Ain Shams University
Faculty of Engineering
Department of Architecture

# MAPPING THE ARCHITECTURAL CONCEPT PROCESS BY UTILIZING ARTIFICIAL INTELLIGENCE IN FORM FINDING

ترسيم عملية الفكرة المعمارية من خلال توظيف الذكاء الاصطناعي في الوصول إلى التشكيل

A Thesis Presented in Partial Fulfillment of the Requirements for Doctor of Philosophy in Architecture Engineering by:

**Abdulrahman Ayman Fahmy Salem**
M.Sc. - Architecture - 2019 – Ain Shams University
B.Sc. - Architecture - 2015 – Ain Shams University

Supervised by:

**Professor Dr. Yasser Mohamed Mansour**
Professor of Theories of Architecture
Ain Shams University

**Professor Dr. Hazem Talaat El Daly**
Professor of Theories of Architecture
Ain Shams University

2024

# Statement

This thesis is submitted to Ain Shams University for the Ph.D. degree in Architecture.

The work included in this thesis was carried out by the researcher at the Department of Architecture, Faculty of Engineering, Ain Shams University, and During the Period from June 7, 2021, to October 2024.

No Part of this thesis has been submitted for a degree of a qualification at any other university or institute.

| | |
|---|---|
| Name | Abdulrahman Ayman Fahmy Salem |
| Signature | |
| Date | |

# Board of Examiners

**Name**: Abdulrahman Ayman Fahmy Salem

**Thesis Title:** MAPPING THE ARCHITECTURAL CONCEPT PROCESS BY UTILIZING ARTIFICIAL INTELLIGENCE IN FORM FINDING

**Degree:** Doctor of Philosophy in Architecture

**Thesis Defense Date: ………/………/…………**

| Board | Signature |
| --- | --- |
| **Professor Dr. Samir Sadek Hosny**<br>Professor of Architecture, Department of Architecture - Faculty of Engineering, Ain Shams University. | |
| **Professor Dr. Yasser Mohammed Mansour**<br>Professor of Theories of Architecture, Department of Architecture - Faculty of Engineering, Ain Shams University. | |
| **Professor Dr. Sherif Morad Abdelkader**<br>Professor of Architecture, Department of Architecture - Faculty of Engineering, The American University in Cairo. | |
| **Professor Dr. Hazem Talaat Eldaly**<br>Professor of Theories of Architecture, Department of Architecture - Faculty of Engineering, Ain Shams University. | |

**Post Graduate Studies:**

Approval stamp

Faculty council approval

The thesis was approved on

University council approval

# Researcher Data

---

**Name**: Abdulrahman Ayman Fahmy Salem

**Job**: Assistant Lecturer at the Faculty of Engineering – Ain Shams University

**Department**: Architecture

**Degree**: Doctor of Philosophy

**Code**: 035983

**Birthday**: 02-02-1993

**Bachelor**: Architecture – June/2015

**Master of Science**: Building Technology – November/2019

**ORCID**: 0000-0002-7258-3431

**E-mail**: abdulrahmanfahmy@eng.asu.edu.eg

# Acknowledgements

***First, of all I would love to thank Allah*** for helping me through the hard times and for rewarding me with such a milestone in my academic life which I hope could help researchers and students locally and worldwide in the future.

***To my beloved father***… This achievement is for you. You have always been there for me whenever I needed support and whenever I needed to talk and express my depression. No words can describe how proud I am to be your son.

***To my beloved mother***, thank you for always being beside me and never giving up on me in the hardest times and for your encouragement and love.

***To my precious wife,*** thanks for being there for me in hard times. Being in my life is a precious gift from Allah.

***To my dear son,*** the love of my life, and Allah's precious gift. You may be tiny now, but you always made me laugh and feel happy whenever I needed it.

I would also love to show my sincere gratitude ***for the dear professor Yasser Mansour***, my mentor and who has always been my source of light every time I felt lost. Without your guidance and knowledge, I would have never reached this place.

***To the dear professor Hazem Eldaly***, this work would have never been completed, without your patience, guidance, and knowledge all the way.

No words can describe how grateful and lucky I am to be a student of yours.

***To my brother, my sister, my dearest friend Amer, my dear beloved Aunt Fawqeya, Ahmed, and Mohammed Essam***, *family, and friends* as well for their encouragement and support.

***And to the great ML engineer Ali Hussein*** for his valuable effort in revising the ML process and code used, without your kind help and patience, I would not have reached this moment. Thank you from my heart.

***And finally, to knowledge… I WANT MORE!***

# Abstract

The rapid advancements in machine learning (ML) have led to numerous practical applications across various fields. Architects and researchers have also begun exploring the potential of ML to enhance their work. However, existing applications often fail to provide precise and readily usable architectural models within the standard design software used by architects. To address this challenge, we present a novel approach that leverages coding geometry in an algorithmic process to translate architectural parameters into machine-understandable data types, such as doubles, integers, and strings. The suggested pipeline starts from modeling a parametric villa on Rhinoceros3d with C# code, creating a large dataset by changing the parameters, and then training ML models with the dataset. The parametric model generated encompasses a wide range of interrelated parameters, including wall dimensions, floor heights, recesses dimensions, window characteristics, building area, setbacks, and land dimensions. The entire model is implemented using RhinoCommon API and C# programming language. The resulting parametric model facilitates automatic storage of data in a CSV file formatted to be used directly in ML. We tested different ML algorithms in this research on four datasets created from the model. A dataset to predict parameters related to the areas, one to predict parameters related to other form parameters, one to predict windows' existence in each wall, and another to predict windows' widths. The datasets require both regression and classification algorithms to predict all the parameters. Impressive results are yielded with ensemble learning methods with all datasets. Regression tasks could reach an R2 score as high as 0.97, 0.79, and 0.99 for areas, other parameters, and windows' widths datasets and 98% accuracy in the windows' existence classification task. All results are computed on the test dataset. These findings highlight the efficacy of this approach in generating accurate architectural predictions through ML techniques.

# Keywords

Black box, architectural design thinking, artificial intelligence, generative design, form finding, machine learning, regression, classification, neural networks, coding, programming.


# List of Acronyms

| Abbreviation | Term |
|---|---|
| ***AD*** | Algorithmic Design |
| ***AdaBoost*** | Adaptive Boosting |
| ***AGI*** | Artificial General Intelligence |
| **Abbreviation** | **Term** |
| ***AI*** | Artificial Intelligence |
| ***ANI*** | Artificial Narrow Intelligence |
| ***ANN*** | Artificial Neural Network |
| ***API*** | Application Programming Interface |
| ***ASI*** | Artificial Super Intelligence |
| ***BERT*** | Bidirectional Encoder Representation from Transformers |
| ***CD*** | Computational Design |
| ***CNN*** | Convolutional Neural Network |
| ***CRNN*** | Convolutional Recurrent Neural Network |
| ***CV*** | Computer Vision |
| ***DL*** | Deep Learning |
| ***DNN*** | Deep Neural Network |
| ***DT*** | Decision Tree |
| ***GAN*** | Generative Adversarial Network |
| ***k-NN*** | k-Nearest Neighbor |
| ***LLM*** | Large Language Model |
| ***LSTM*** | Long Short-Term Memory |
| ***MAE*** | Mean Absolute Error |
| ***ML*** | Machine Learning |
| ***MLP*** | Multi-Layer Perceptron |
| ***MSE*** | Mean Squared Error |
| ***NLP*** | Natural Language Processing |
| ***PCA*** | Principal Component Analysis |
| ***PD*** | Parametric Design |
| ***ReLU*** | Rectified Linear Unit |

| | |
|---|---|
| ***RF*** | Random Forest |
| ***RNN*** | Recurrent Neural Network |
| ***SDK*** | Software Development Kit |
| ***SVC*** | Support Vector Classification |
| ***SVM*** | Support Vector Machine |
| ***SVR*** | Support Vector Regression |
| ***VAE*** | Variational AutoEncoder |
| ***VPL*** | Visual Programming Language |
| ***XGBoost*** | eXtreme Gradient Boosting |

***For a Glossary: Please, refer to appendix A.***

# Contents

# List of Figures

# List of Tables

## I. Overview

Architectural design methods commonly employed by architects are often regarded as vague and difficult to formalize. Among these is the "black box" design approach, where the generation of design concepts occurs solely in the architect's mind without external articulation. This approach aligns with the desire of many architects to express individuality, resulting in the development of a personal architectural language or style. However, this individual expression can lead to a degree of repetitiveness, as architects often reproduce similar forms that reflect their personal design philosophies (Schön, 1984). Furthermore, the design process itself is inherently non-linear. It follows a cyclic, iterative pattern in which an initial concept undergoes continuous development and refinement through the architect's critical analysis at each stage of the design process. This iterative nature complicates the explanation and formal documentation of how the design process functions in practice (Lawson, 2006).

The advent of the digital era has had a profound impact on various fields, including architecture. Initially, architectural design was conceptualized through manual sketches on paper. Today, however, architects rely on a wide range of digital tools and software to translate increasingly complex ideas, which have been made possible by the development of new materials and fabrication techniques. These advancements have facilitated the construction of free-form buildings, once difficult to realize through traditional methods (Kolarevic, 2003). The digital revolution did not stop at merely aiding architects in expressing their ideas; it brought about a significant paradigm shift with the introduction of generative design. In generative design, software generates numerous design alternatives based on input variables, constraints, and algorithms. This allows the architect to explore a vast array of potential solutions, as opposed to manually developing only a few alternatives (Mitchell, 2005).

This shift has prompted critical debate within the architectural community. Some argue that manually generating a small number of well-considered design alternatives is more efficient and professional than sifting through thousands of machine-generated options, many of which may lack coherence or thorough analysis (Oxman, 2017). Moreover, the sheer volume of alternatives produced through generative design can overwhelm architects during the decision-making phase, delaying the selection of the optimal design solution (Duarte, 2001).

However, what if machines were able to produce fewer, but more refined, alternatives? By incorporating architect-specific variables, constraints, and algorithms—reflecting the architect's design logic—it might be possible to generate fewer, but well-studied, design solutions. Additionally, applying artificial intelligence (AI) and machine learning (ML) technologies could enable the machine to "learn" from the architect's preferences, decisions, and design style, allowing it to generate outcomes that align more closely with the architect's intentions (Gero & Kannengiesser, 2014).

AI has seen rapid advancements across many industries, with an increasing number of tools becoming publicly accessible. In the field of architecture, generative AI applications are being explored for a range of tasks, from conceptual design and visualization to automatic generation of plans (Burry, 2016). However, generative AI technologies like point-clouds, voxels, or NeRF models, while effective for visual outputs, often fail to produce architectural models that are clean and developable. There are ongoing experiments with non-generative AI applications for tasks such as material prediction, classification, and urban planning (Peters, 2013).

This research explores the potential of AI and ML in architectural design, specifically in form-finding and form-making processes. A critical aspect of this investigation is the translation of architectural style and vocabulary into quantifiable variables and constraints that can be processed by AI to generate designs that not only meet technical requirements but also align with the architect's creative vision. Moreover, the integration of coding and algorithms in this framework is analyzed, as it provides a more structured approach to utilizing ML in the design process. Several ML models are tested and evaluated to determine their suitability for these applications, contributing insights into the development of AI-assisted design frameworks.

## II. Problem Definition

Generative design, as beneficial as it might seem, has some shortcomings, especially in the decision-making phase. In this process, thousands of variations are produced so that the architect could choose a suitable design to develop. However, many of these variations might not be reliable nor logical depending on many aspects like building codes, architectural style, human behavior, spaces relations, or even architects' aesthetic sense and artistic preferences regarding form making. In this manner, architects might consume a lot of time trying to sort out the best generated ideas for further studies and design development. In addition, the more the architect provides the program with constraints and variables, the more ideas the program will generate leading to more time consumption in the decision-making phase.

On the other hand, the machine needs a lot of information to sort out the best alternatives and ignore the ones which do not imply the architect's character which shapes his own black box of ideas, id Est, optimizes the results and so that has a role in the decision-making process leading to a smaller number of variations with a better design quality rather than a larger quantity.

The time spent designing a building takes a lot of manhours. Even designing more prototypes of the building will still take a lot of time although the style may be already determined.

To utilize AI in time saving, some generative AI models which generates photos (designs?) of plans and perspectives is present and is being used extensively by architects nowadays. However, such applications come with many issues regarding authenticity, creativity, and even functionality of the result which we argue should not be even considered an architectural product.

Finally, to automate some tasks in the design phase, some classification and regression applications have been experimented to predict or classify materials, architectural parameters, etc. However, we could not trace any trials to create architectural form models through ML which could be directly used in other phases such as schematic design and design development.

Although AI and ML field is considered highly developed today, many concepts in this field need to be revisited by architects to harness the power within the machine in automating time-consuming tasks.

## *III. Research Aim*

This research aims at utilizing ML in decision-making so that the machine could learn from previous decisions and lower down the probabilities of the resulting architectural forms by taking on the architect's character resulting in models and alternatives close to what the architect could design with a normal black box design approach. In this sense, the idea of getting the machine to decide values of architectural parameters which forms a building model through a framework which utilizes ML is the main aim.

The aim of this research could be achieved through the following objectives:

- Defining the architectural design process, and architectural design thinking.
- Defining metaphors in architectural thinking including black box, glass box, and gray box.
- Investigating and mapping the vocabulary and elements of contemporary architecture style.
- Applying certain relations between the parameters to form generative models through coding.
- Investigating AI and ML applications to understand their capabilities and decide how to benefit from them.
- Defining a framework for decision-making using ML.
- Driving a methodology framework for utilizing ML to generate designs that are relevant to the architect's/project's patterns and previous choices and preferences took by the architect in this regard.

## *IV. Research Hypothesis*

An architect's way of thinking is considered a process that could be traceable in the conception thinking phase (grey box). This process coupled with the vocabulary of an architectural style could be translated into parameters that shall combine to produce the result of what occurs in the mind of the architect in this very process. If these parameters are well studied and well introduced to the machine through 'coding', the machine could provide relevant and precise generative designs that could speed up decision-making. In addition, the machine could learn from the architect's choices and thus give more precise and more relevant designs either in the next phase or even the next project. This ML mechanism could take architecture to a new era of human-machine interactive architectural conceptualization.

## *V. Literature Review*

This review briefly shows the previous work and research done considering architectural design methods, contemporary architecture, generative design, algorithmic design, and applying AI in architecture. The review includes investigations about different design methods including the black box and the glass box, explanation of architectural design literature, explorations in contemporary architecture motifs and finding its patterns, generative design, and algorithmic design as design methods with its pros and cons, and different applications and terminology of AI and ML in architecture field.

- ***Previous literature regarding architectural design methods:***

  Since the Greeks, propositional knowledge which asks epistemological questions about the evidence of asserted claims and truth, or falsity has been the focus of western philosophical studies. Design problems which are hard to define according to Lawson (1980) relate to various epistemological questions and while designers search for answers in the design process, they contribute to the interpretation of a design problem. In this process, designers could modify the rules when processing the information leading to a paradigmatic revolution where either a new entity arises, or an entire system falls (Rittel, 1972).

Architectural design has seen many attempts to be defined. It is defined in terms of certain fields of knowledge as Rowe's thoughts of it as being located in an ambivalent position between technical science and forms of fine art (Rowe, 1987). Vitruvius defined design as providing three main factors which are firmness, commodity, and delight. However, the accepted theory of successful designs evaluation is more likely based on Vitruvius' three factors, and this is seen in contemporary linguistic studies using terms which are used by Vitruvius like semantic, pragmatic, and syntactic.

Architectural design as defined above has five important components that designers should consider throughout design process phases. These components are aesthetics, culture, environment, structure and materials, and economics and social influence. Also, it has methods which include six approaches: "black box", "glass box", problem structure, control, observation, and evolution (Broadbent, 1969). Studying these methods is important to recapture the activities involved with design decision-making so that architects could follow defined procedures from formulating a program to achieving an effective and efficient final solution.

In the "black box" approach, mystery and creativity are the two main characteristics of the design process. The process usually occurs in the designer's mind therefore it is hard to analyze the design. However, techniques like brainstorming and applying synetics could help to visualize the process itself.

On the other hand, the "glass box" approach, design is analyzed based on the logical process and decision sequence of the design. The process in this case is a sequential events entity including identification, analysis, synthesis, and evaluation. According to Broadbent, (1969), this approach could be applied through some methods including operational research, critical path, systems analysis, set theory, logical model, “feed-forward”, and design territory map. Also, in this approach, two distinct design structures are observed which are a sequential process such as the sequential structure of the design process with its twelve major chronological phases included in the handbook from the Royal Institute of British Architects (RIBA) and an iterative/cyclic process where a feedback loop before the completion of the project is attached to different phases as in Levin’s eleven stages in the decision sequence. This approach has seen criticism because it focuses on art development rather than concerning for actual buildings’ practical aspects.

The problem structure method is composed of many variations including morphological analysis, inter-connected decision areas, decomposition analysis and relational theory. According to Aismow (1962), the design process is divided into seven phases: feasibility study, preliminary, detailed design, planning for production, consumption, and retirement. In the preliminary design, a best design is identified from a number of alternatives.

However, an evolution of design methods appeared after Popper’s systems approach to problem solving as well as his philosophy of science applying the deductive method of testing. Therefore, architectural design methods could be described as an iterative process based on trial-and-error which relies on experience, knowledge, and intuition. And so, According to Rzevski, 1980, design process has four features: investigative, creative, rational, and decision-making process. This problem-solving framework involves four steps which are problem understanding, tentative solution generation, iterative testing and refining of details, and finally, design solution outputting. This solution suggests new design problems in the future.

- ***Previous literature regarding generative design and algorithmic design:***

  Dorst, K., 2003, explains how design problems in architecture are open-ended, ill-structured, and unique and how solving such problems requires problem-specific and experimental methodology. In this manner, architects are not allowed to solely rely on predefined methodologies nor approved solutions to similar problems. Architectural problems are complex and have a wide range of subconscious factors on various levels that range from building codes to aesthetic aspects. The complexity of such problems requires the reasoning, guessing, and intuitive decision-making of an architect.

  In recent years, generative design has evolved when form-finding techniques were introduced through computational tools. This approach revolutionized architectural design and production where new design paths were offered to architects favoring computationally generated complexities over predictable relationships between form and presentation. Thus, the emphasis was shifted from “form making” to “form finding” (Kolarevic, 2003).

  According to Agkathidis, A., 2015, generative design could be described as a method by which the form is generated based on rules or algorithms which are often derived from computational tools as processing and scripting platforms. Being influenced by Jacques Derrida’s deconstruction theory, Peter Eisenman applied design techniques such as fractals, scaling, overlay, and superposition in relation to rules of order and thus designed several projects on this basis. This could be thought of as the first contemporary generative design attempt before the advent of digital architecture and various software which offered new possibilities.

  Agakathidis, A., 2015, briefs generative design techniques as:

1- *continuous surface (soft mesh, double-curved shells, and hyperboloids)*

2- *Modularity and accumulation (interlocking units and irregular units)*

3- *Deformation and subtraction (twisted block and porous space)*

4- *Algorithmic patterns (tessellated planes and Voronoi surface)*

5- *Triangulation (3D Penrose pattern and faceted loft)*.

Generative design is usually criticized for disconnecting the output from its context and users. This could lead to decreasing spatial quality and integration of the building within the urban environment. In addition, it is criticized for disconnecting the architect from drafting techniques and physical modelling which once formed the essential foundations of architectural education, risking the loss of material effects and properties.

Generative algorithms are defined as parametric ways that could handle geometry in design problems (Khabazi, 2012). Using this type of algorithms, designers could utilize a lot of possibilities regarding geometric computing and also manage large amounts of data and calculations easier than conventional geometry methods. This approach is not only limited to predetermined experiments but rather serves the exploration of unlimited potentials. (Gunagama, 2017).

Maldonado, 2014, describes an algorithm as a cooking recipe with a step-by-step guide. According to this description, an algorithm in architecture requires limitations in design which are analogous to the variables and parameters in a cooking recipe. Therefore, variables and parameters are necessary for an algorithm system. In this context, variables are entities which change in the system while the parameters are entities that are used to unify or connect various variables of an equation (Gunagama, 2017). Gunagama concludes that despite of the various alternatives that could be developed through generative design, the breadth of the definition of optimal design as well as the lack of ‘the ability to translate verbal ideas to mathematical’ could lead to limitation in the resulting alternatives.

Also, according to Singh, V., 2012, adopting generative design (GD) systems in architecture is meant to support human designers through computational capabilities as well as automate parts of the design process. The commonly used generative design techniques according to him are shape grammars, L-systems, cellular automata, genetic algorithms, and swarm intelligence. Also, most of the existing generative design systems are derived from one of these techniques.

In addition, Krause, J., 2003, experienced the generative design process in architecture, and described it as a teaching process where the architect is a teacher, and the computer is the student. Krause claimed that a person cannot really understand something until he teaches it to a computer. He described the process steps as follows:

1- *Start with a goal.*

2- *Describe consistencies.*

3- *Formalize code parts.*

4- *Set range potential.*

5- *Evaluate output.*

6- *Add complexity.*

7- *Increase tectonic potential.*

8- *Iterate.*

- ***Previous literature regarding contemporary architecture:***

In 1929 Hugh Ferriss published his book The Metropolis of Tomorrow. He presented designs of various functions imagining the city of the future. In one of his drawings, Night in the Science Zone, he presented a skyscraper without any details rising from amidst the houses surrounding it. He included a poem to this drawing: “Buildings like crystals. Walls of transparent glass.

Ordinary glass hollow bricks covering the steel grid. Without Gothic art: without acanthus leaves: without memories of the plant world. The mineral kingdom. Glittering stalagmites. Forms as cold as ice. Mathematics. Night in the Science Zone." These words formed later the manifesto of the future generations of expressionist architects.

According to Kozlowski, 2013, the advent of contemporary expressionism in architecture has seen light through Zaha Hadid's design of the Peak Hong Kong Club in 1983. All of the trends like cubism, futurism, formism, etc. are considered the languages of 'expressionism'.

Hohenadel, K., 2020, answers the question "what is contemporary architecture?" as the current style of architecture where building built according to current trends in a time would be considered contemporary architecture. The author claims that the characteristics and elements of contemporary architecture include curved lines, rounded forms, unconventional volumes, asymmetry, free-form shapes, open floor plans, large windows, green roofs, living walls, integration into the surrounding landscape, integrated smart technologies, and integrated customizable LED lighting. In addition, flat roofs, geometric simplicity, open spaces, environmental considerations, and volumes compositions could be considered motifs of the contemporary architecture.

Reffat, R., 2008, investigated patterns of contemporary architecture in Saudi Arabia using data mining (DM) techniques. Reffat suggests that every place gains its character by certain patterns of events which are not necessarily human events. The elements of the building (walls, windows, rooms, doors, etc.) repeat a lot, but they vary every time they occur. Hence, the fact that the elements themselves vary says that they are not the repeating events. The events in this case could be the patterns of relationships between the elements. Each of these patterns is a three-part rule expressing the relation between context, problem, and solution. The author claimed that traditional methods of data analysis including spreadsheets and ad-hoc queries were capable of only creating informative reports from data and could not analyse the contents of these reports and thus not adequate. So, he used data mining which is a process which discovers patterns and relationships in data which may could be used to make valid predictions through a variety of data analysis tools. Data mining functions were performed including summarization, association, classification, prediction, and clustering. The studied building characteristics regarding form and facades were the organization of the building (centralized, linear, radial, cluster, or grid), building orientation, building height, main entry façade direction, fenestration pattern, shading devices, glazing, building envelope, external finishes, and façade style. A lot of characteristics were not studied including the dimensions of glazing, repetitive motifs, and form generation variables. The data mining process includes data pre-processing where noisy and incomplete data are removed, data transformation where data is stored in various tables, and data warehousing, which is the process of visioning, planning, building, using, managing, maintaining, and enhancing data bases. Data could be stored in WEKA's data warehouse which is composed of ML algorithms for solving real-world data mining problems. Reffat used WEKA and IBM intelligent miners for mining the data.

- ***Previous literature regarding AI and ML:***

On the other hand, the advent of AI in the world has affected a lot of fields including architecture. Applying AI in the architectural discipline has been investigated in many research and experiments recently.

Chaillou, S., 2019, studied the application of AI to floor plans generation and analysis. The goal took the usual sequence of AI applications in the architectural field which has 3 steps: generation, classification, and presentation where users can browse through the generated design alternatives. However, their study dealt with the design process as a sequential process with successive design steps which contradicts with the cyclical nature of the design process where a designer thinks, analyses, develops, and analyses again until a satisfying and problem-solving solution appears. The researcher used two of AI's main fields of investigation in their study: analytics and generative adversarial neural networks (GAN). The GANs -as any machine-learning model- could learn statistically significant phenomena among data presented to them. However, their structure is made of two key models which are the Generator and the Discriminator. GANs could generate a loop between the two models to refine their relevant images generation ability. The Discriminator works to recognize images from a set of data. On the other hand, the Generator works in creating images which resemble images from the same dataset. In this study, the machine took 250 iterations to be able to build some sort of intuition for itself after being trained for a day and a half. Before those iterations, the initial attempts were imprecise. In order to qualify the results, 6 aspects of the floor plan design were used as metrics including footprint and orientation. Each metric is translated to numbers, colours, or matrices to establish a proper communication with the computer about its characteristics and shape. From this study, the qualifying or classification phase is a crucial phase in AI applications because the finer the metrics are provided, the better and higher will the quality of the generated options be.

Also, Zheng, H., 2018, and Martinez, N., 2017, investigated GANs as design assistants where they studied the idea of creating a loop between the designer and the machine to refine the design process.

According to Malaeb, J., and Ma, W., 2019, the developed understanding of how human brain works led to changes in the concept of AI where a machine focuses on mapping human behavior rather than only carrying out complex calculations and working as a memory. AI mostly works with deep learning and natural language processing technologies. By processing large amounts of data and classifying it based on patterns recognition, machines could be trained to accomplish specific tasks. The authors mention that various attempts to introduce AI to the architectural field exist. However, most of the trials are partially applying AI because they require human intervention and monitoring to do a major part of the work. Again, the authors confirm a main concern with AI where data inaccuracies always reflect on the results. That is why limitations of AI in creative fields are obvious where a machine learns from data. In addition, the authors tackled deep learning and explained that it involves feeding the machine with a lot of data that could help making decisions about other data. The data is fed through neural networks which could extract numerical values of the data which pass through them and then classify the data according to the answers received.

Also, Bishop, 2006, defines ML as a subset of AI. It is "the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead." A mathematical model based on sample data known as "training data" is built by ML algorithms. This model helps to make predictions or decisions without being explicitly programmed to perform the task.

Last but not least, As, I., et. al., 2018, tackled the idea of applying deep neural network (DNN) to extract design into essential building blocks based on functional performance criteria and then recombining them into new designs. The idea is based on a historical event from the 16th century in Spain where the Italian architect Giacomo Barozzi da Vignola was hired by the king Philip II to initiate a competition to design a monastery in Madrid. 22 architects submitted their entries, but Vignola composed a new design from the 22 submitted designs by collaging bits and pieces instead of choosing a winning project. However, the king was impressed although the idea does not seem to be ethical in today's profession's morals. The authors investigated two ML methods for design generation which are DNNs for convolution and representation learning and GANs. In the first method, a graph convolutional neural network is used to discover essential building blocks which respond to certain functional criteria. After that, the building blocks are merged into new designs with the use of graph-theory methods and data about the proximity of nodes in latent vector embeddings. However, DNNs are not set up to generate new designs, instead, they are very effective at classification and discovery. The second method (GANs) is considered a new version of DNNs which the authors used to merge building blocks into new compositions. At the end, the researchers confirmed the initial validation of graph-processing DNNs in generating novel conceptual designs although some limitations and constraints were faced including the complexity of architecture field (design scope) where they only focused on the function and not the aesthetical and structural aspects, design data work where some design samples could not be labelled nor converted to graphs on Revit API's BIM format, and the evaluation of generated designs where the authors did not deal with quantitative evaluation of the new compositions. As, I., et. al., 2018, also suggest non-manifold topology as an alternative method to graph presentation which could allow the representation of walls, corridors, and enclosed spaces by topological objects like faces, shells, and cells.

## *VI. Research Methods and Tools*

The research goes through different successive methodologies to achieve its aim and objectives. These methodologies are:

| Methodology | Application |
|---|---|
| ***Critical Analysis*** | Defining thinking and problem-solving |
| | Defining architectural thinking |
| | Investigating complexities in architectural design |
| | Analyzing the design process and form generation |
| | Mapping contemporary architecture patterns and motifs |
| | Investigating generative design and algorithmic design |

| | |
|---|---|
| | Investigating AI and ML definitions, types, and applications |
| ***Case Studies*** | Analyzing AI and ML applications in architecture |
| ***Experiments*** | Turning the elements of a building into parameters through algorithms building. |
| | Writing a generative design program through coding |
| | Generating a parameters data set that could be used to train ML models |
| | Identifying the problems and deciding the proper ML models and tasks |
| | Applying ML so that the machine could learn from the architect by mapping the data set and predict/classify the parameters of the building under study |
| | Applying coding to fine-tune the resulting 3d-model so that the architect can interfere in both teaching the machine and modifying the result easily. |

## *VII. Research Scope & Limitations*

This research is directed towards investigating ML supervised learning models specifically from AI models. ML models could predict, classify, or cluster data based on the data sets they learn from. So, models do not ‘generate’ new data but generalize to unseen data based on mathematical concepts from linear algebra, numeric methods, and optimization. Supervised learning is chosen so that the machine can predict numbers and classes which map to architectural parameters defined by the architect as the framework targets a human-centered design approach. Other applications like clustering could not help with the intended product.

However, a wider spectrum of AI models is investigated to understand and analyze the differences between generative and discriminative AI in terms of concepts and applications.

In addition, as this type of application’s result could be considered ‘generative design’, analysis of how generative design systems work is necessary to gain insights about how ML applications could be different.

To work with ML, it is recommended that architectural design modeling be done through coding, so, the research is also directed towards a thorough study on how to harness the power and freedom of coding in form generation. The suggested framework allows the architect to create design options as samples used for training. The architect's designs (data set) totally depend on their choice. In this case the architect should judge those designs themselves because whatever the data set looks like, if a pattern exists between the building requirements and target building parameters, the framework should work as intended, and the accuracy of ML models should be high. To test the framework, a contemporary building is designed and modelled through coding in C# in Grasshopper3d for Rhinoceros3d 7 using RhinoCommon API which were chosen for the fast interface and processing of algorithms. Other software could have resulted in the same result.

For simplicity in the framework's validation stage in this research, the result from this first step is a data set containing tangible form proportions-related parameters values of 600 prototypes of the contemporary villa consciously designed by the author (an architect) based on some cases related to the total built-up area which ranges between 200 and 1000 $m^2$, the neighbor types, setbacks, land dimensions, and other scenarios. The villa was chosen to have a 'contemporary' style for ease of geometry in building and coding as the main concept is to find a relationship between almost a hundred of parameters of the villa, and to train the ML system and test if it could find a pattern between them. For the sake of simplicity, the location of the villa (country) was not included in the equation.

Different ML and ANN algorithms were tested for both classification and regression tasks which were assigned to either predict values of the parameters or to classify some parameters and achieve the aim of the research.

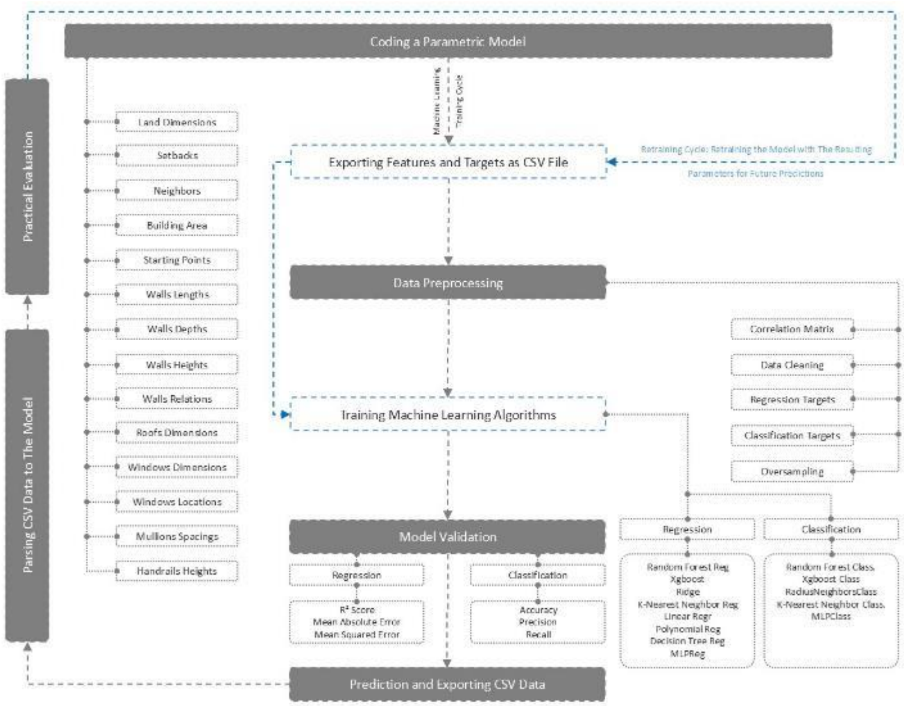
## *VIII. Research Structure*

| Part | Chapter | Chapter Subtitles | Methodology |
|---|---|---|---|
| **Part 1: Architectural Design Process from Architects to Machines** | **Chapter 1:** Architectural Design Thinking and Process | *1- Thinking and Problem-Solving*<br>*2- Architectural Design Thinking*<br>*3- The Black Box vs the Glass Box vs the Gray Box Metaphors*<br>*4- Design Methodologies*<br>*5- Architectural Design Process*<br>*6- New Technologies Effects on the Design Process*<br>*7- Complexities in Architectural Design*<br>*8- Evaluation of Architectural Design Outcome* | *Critical Analysis and Case Studies* |
| | **Chapter 2:** Mapping the Elements of Forms in Architecture | *1- Characteristics of Architectural Forms*<br>*2- Form Generation in Architectural Design*<br>*3- Mapping and Analyzing the Elements and Motifs of Contemporary Forms* | |

| | | | |
|---|---|---|---|
| | **Chapter 3:**<br>Coding in Computational Design: A Base for Utilizing AI in Form Finding | *1- Computational Design thinking*<br>*2- The Roots of Computational Design*<br>*3- A Taxonomy of Computational Design Terms*<br>*4- Generative Form Finding*<br>*5- Architectural Forms as Information*<br>*6- Coding as a Practice*<br>*7- How Modeling Software Work*<br>*8- Visual Programming Languages*<br>*9- Bias in Modeling Processes and Leveraging Power, Freedom, and Spruceness of Coding* | |
| **Part 2: Integrating AI in the Architectural Design Process** | **Chapter 4:**<br>Artificial Intelligence and Machine Learning in Architecture | *1- AI Definition and History*<br>*2- Types and Applications of AI*<br>*3- Generative and Non-Generative AI*<br>*4- Machine Learning Definition and Types*<br>*5- Data Sets in Machine Learning*<br>*6- Machine Learning Algorithms*<br>*7- A Review on Using Non-Gen AI in Architecture*<br>*8- A Review on Using Gen AI in Architecture*<br>*9- Generative AI Drawbacks in Architectural Design*<br>*10- Generative and Non-Generative AI Usage Possibilities in Architectural Design* | *Critical Analysis and Case Studies* |

| | | | |
|---|---|---|---|
| | **Chapter 5:** Architectural Form Generation: Applying Machine Learning Algorithms on Architectural Parameters Data Sets | *1- Problem Definition, Scope, And Limitation*<br>*2- Materials and Methods*<br>*3- Coding an Architectural Design Model*<br>*4- Generating a Machine-Learning Ready Data Set*<br>*5- Data Correlations*<br>*6- Data Pre-Processing*<br>*7- Data Splitting and Choosing Features and Targets*<br>*8- Data Resampling*<br>*9- Train-Test Splitting*<br>*10- Training Models* | *Experiments* |
| | **Chapter 6:** Machine Learning Analysis and Results | *1- Feature Importance*<br>*2- Evaluation Metrics*<br>*3- Model Learning Analysis*<br>*4- Predictions*<br>*5- Model Fine-Tuning*<br>*6- Discussion and Conclusion* | |
| ***Research Conclusion and Future Direction*** | | | |

# I. Framework of the Study




*Suggested Machine Learning Framework for Regression and Classification Tasks in Architectural Design Modeling - By the Author*

# Part 1: Architectural Design Process from Architects to Machines

# Chapter 1: Architectural Design Thinking and Process

## Preface

Architectural design is a very complicated process which exhibits a very large number of parameters and possibilities. In his book “How Designers Think”, Bryan Lawson argues that the essence of design necessitates different ways of thinking. As the author describes, thinking could be either in closed systems or adventurous. But with all the parameters that affect architects’ decisions and ways of thinking, how is it possible to map every idea that sparks in the designers’ brain or even the design process? Those parameters can be quantitative as the legal constraints, technical aspects, financial aspects, and clients’ desired number of rooms in a building. And they can be qualitative including architectural theories, psychological factors, architects’ philosophical approaches, and aesthetical aspects which usually depend on the designer’s favored architectural style, language, and motifs.

Architectural innovation has always drawn inspiration from the concealed depths of black boxes as well as the transparent clarity of glass boxes. And the design thinking process in architecture, often considered a vague merger of intuition, expertise, and inspiration, has been metaphorically likened to a black box – a mysterious entity in which decisions are made and ideas take shape, shielded from external scrutiny. Yet, counter to this notion is the transparent principles of the glass box, where design decisions are laid bare for all to witness, fostering collaboration, critique, and shared understanding.

This chapter explores what architectural design thinking is through an investigation of different design thinking styles and design types as well as how to define good architecture. Different complexities in design thinking are explained leading to the important questioning of whether architectural design thinking should be encapsulated in a black box or a glass box after explaining the two metaphors. This chapter is an invitation to critically reflect on the metaphors that define the understanding of the architectural design process. It challenges preconceived notions, encourages thoughtful examination, and paves the way for a deeper comprehension of the multifaceted nature of design thinking in architecture.

Design is a strategic approach to problem-solving, leveraging creative abilities that integrate elements from both the arts and sciences to address diverse challenges. While designers employ various problem-solving methods, they typically adhere to an established pattern or sequence of steps that have proven effective in realizing designs from the initial concept to their completion. Whether performed consciously or subconsciously, the design process is a fundamental aspect integral to nearly every project.

In the realm of architecture, the design process and methodology play a crucial role in crafting innovative solutions. Numerous studies have delved into design methodologies, engaging in critical analysis, evaluation, comparison, and the proposition of alternative approaches for creative problem-solving. These studies serve a dual purpose: assisting designers in understanding their distinctive styles and presenting novel options for achieving solutions.

The stimulus for such research lies in recognizing that each architect operates within their unique conceptual frameworks. These frameworks not only aid in setting boundaries and defining objectives but also enhance communication within extensive networks of collaborators. In summary, the passage underscores the structured yet creative essence of the design process and emphasizes ongoing efforts to refine it through methodological exploration.

In this chapter, the term architectural design process is explained, and different methodologies are discussed. Also, the effects of other disciplines on the process are discussed as well as the effects of today's technology on different architectural design processes.

## 1.1 Thinking and problem-solving

Understanding the design process requires insight into human cognition and thinking, as highlighted by Lawson (1990) and Caldwell, et. Al.

(2000). Psychologists believe creativity is linked to brain function and neural processes, with individuals varying in their cognitive wiring. Since ancient times, the debate over the origins of knowledge—whether it is acquired through experience (empiricism) or inherent (nativism)—has intrigued thinkers like Plato and Aristotle. Rowe (1987) identifies two themes in problem-solving: one grounded in mental processes governed by lawlike relationships and the other in behavioral, non-mentalistic terms. The evolution of design thought in architecture has been shaped by five key psychological perspectives: Associationism, The Wurzburg School, The Gestalt Movement, Behaviorism, and Cognitivism.

Associationism viewed creative problem-solving as mechanistic and atomistic, while The Wurzburg School, led by Kulpe, emphasized task-oriented act theories, influencing principles like Sullivan's "form follows function." The Gestalt movement introduced holistic principles, emphasizing whole units in perception and problem-solving, impacting architectural design through comprehensive images. Behaviorism focused on observable behavior and stimulus-response models, influencing practical approaches like climate-responsive design. Cognitivism, integrating Gestalt psychology, views the mind as an information processor, relevant in the modern context due to the complexity and volume of information in contemporary design projects (Mahmoodi, 2001).

Thinking styles are diverse and categorized into problem-directed, undirected, and creative thinking (Gilhooly, 1996). Directed thinking involves solving well-defined problems using state-action or problem reduction methods. Undirected thinking, such as daydreaming, often occurs in the pre-concept phase, potentially leading to innovative design ideas. Creative thinking, characterized by Wallas (1926) in four phases—Preparation, Incubation, Illumination, and Verification—generates novel and valuable products. Styles of thinking, defined as preferred ways of thinking (Mahmoodi, 2001), vary among individuals, with societal perceptions of capability influenced by the fit between thinking styles and tasks. Exploring these styles enhances the understanding of how architectural designers approach problem-solving.

Table 1-1 summarizes thinking styles using Sternberg's (1997) metaphor of governments, which have diverse functions (legislative, executive, judicial), forms (monarchic, hierarchic, oligarchic, anarchic), levels (global, local), orientations (external, internal), and leanings (liberal, conservative). Similarly, styles should consider these various aspects of individual functioning.

*Table 1-1 Summary of Styles of Thinking (Sternberg, 1997, p. 27)*

| Functions | Forms | Levels | Scope | Leanings |
|---|---|---|---|---|
| Legislative<br>Executive<br>Judicial | Monarchic<br>Hierarchic<br>Oligarchic<br>Anarchic | Global<br>Local | Internal<br>External | Liberal<br>Conservative |

Mental self-government has three functions—Legislative (creating rules), Executive (following rules), and Judicial (evaluating rules)—and manifests in styles akin to government forms: monarchic (focused), hierarchic (priority-setting), oligarchic (balancing goals), and anarchic (creative). These styles vary by level, with global thinkers addressing abstract issues and local thinkers focusing on details; by scope, with internal thinkers being introverted and task-oriented, and external thinkers being extroverted and people-oriented; and by leaning, with liberals seeking change and conservatives preferring structure.

Since classical times, it has long been recognized that the human brain possesses two distinct modes of thinking and understanding (Table 1-2).

*Table 1-2 Two Types of Thinking described by intellectuals (URL-2, CaldwelJ, et al., 2000)*

| | Left Brain | Right Brain |
|---|---|---|
| **Maslow** | Rational | Intuitive |
| **Bruner** | Rational | Metaphoric |
| **Koestler** | Associative Thinking | Bisociative Thinking |
| **De Bono** | Vertical | Horizontal or Lateral |
| **Bronowski** | Deductive | Imaginative |

| | | |
|---|---|---|
| **Shopenhauer** | Objective | Subjective |
| **Freud** | Secondary Process | Primary Process |
| **Jung** | Causal | Acausal |
| **Langer** | Discursive Symbolism | Presentational Symbolism |
| **Neisser** | Sequential Processing | Multiple Processing |
| **Kubie** | Conscious Processing | Preconscious Processing |

The human brain consists of two cerebral hemispheres (Figure 1-1), with the left hemisphere typically dominant. The left hemisphere is slightly larger, and the two are connected by the Corpus Callosum. Roger Sperry's split-brain experiments in the 1950s revealed distinct functions for the left and right hemispheres, each controlling the opposite side of the body.

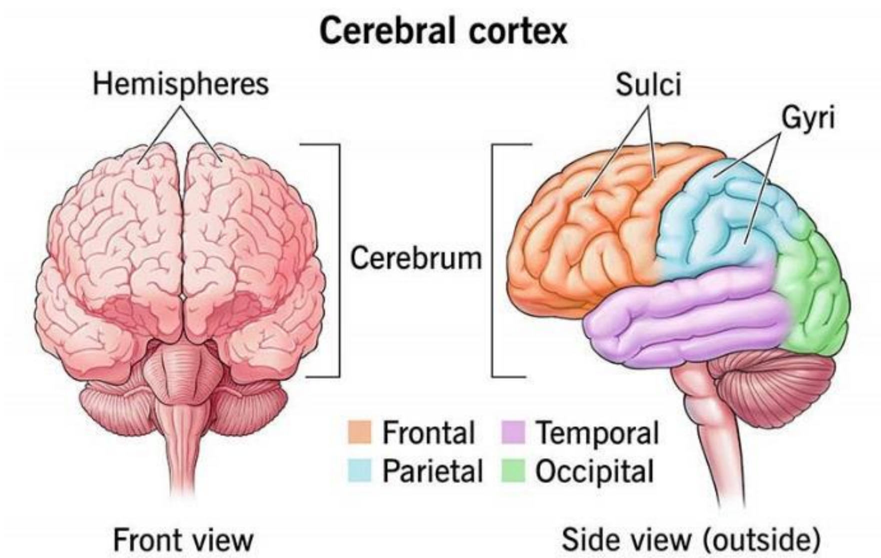



*Figure 1-1 Cerebral Cortex (Human Brain) - https://my.clevelandclinic.org/health/articles/23073-cerebral-cortex (Last Access: 14/11/2023)*

Different brain regions process information differently, with visual word perception and word meaning involving distinct areas. Human actions and thoughts are controlled by one dominant hemisphere. Initially, hemispheric differences were thought to be based on behavior modality, with the left hemisphere specialized for language and reasoning, and the right for music and vision. This evolved into a distinction between a "rational" left hemisphere and an "intuitive" right hemisphere. More recently, discussions emphasize a "left-analytic" and "right-holistic" mode of information processing, suggesting each hemisphere has a distinct cognitive style. The left is associated with sequential, analytic thinking, common in Western societies, while the right is linked to holistic, intuitive thinking, more prevalent in Eastern cultures and religions (McGilchrist, I., 2009).

The present discussions on hemisphere characteristics should not imply that thinking activities are exclusively determined by either the left or right hemisphere. The design process benefits from the interaction between the two hemispheres, leading to well-rounded thoughts and actions.

According to Tovey (1984), designing and problem-solving engage both hemispheres by matching analytically processed problem models with holistically processed solution patterns. Evidence indicates that both hemispheres contribute in parallel during higher-level mental activities, exchanging information. Both types of thinking are crucial in addressing design problems, but the dominant hemisphere may vary based on the adopted strategy.

Understanding how the human brain thinks remains a crucial subject that helps us to understand how any thinking or problem-solving process occurs in the human’s mind. From a further perspective, understanding how architects think and solve problems will be a very important discussion in the coming years because of the spread of AI applications in architectural design recently which will have higher chances to be developed once frameworks of how architects think are developed, AI can learn from these frameworks and act accordingly.

## 1.2 Architectural design thinking

Architectural design involves solving diverse problems, including site issues, social effects, space planning, construction technologies, environmental aspects, legal constraints, and cost. This process requires thinking skills like analysis, synthesis, and evaluation, encompassing both well-defined and ill-defined aspects. Not all problem-solving thoughts are documented; some remain in the architect's mind until a complete solution emerges. Definitions of architecture vary, often blending art and science. Influential figures like John Ruskin, Le Corbusier, Louis Kahn, and Norval White emphasize both the physical and mental aspects of architecture. Ruskin and William Morris view architecture as Building + Art. Conway and Roenisch (1994) note its Greek origin meaning 'builder,' yet architecture is broader, affecting social, cultural, and economic aspects. Capon (1999) categorizes architecture using Aristotle's six categories: Substance (construction), Relation (context), Quantity (form), Quality (meaning), Acting, and being Acted upon. This multi-disciplinary identity shapes lives and societies, requiring characteristics like uniqueness and positive psychological impacts. Capon organizes the six elements of good architecture into two main categories as in table 1-3.

*Table 1-3 - Capon's categorization of the six elements of good architecture (Capon, 1999a, P. 181)*

| **Primary Categories** | |
|---|---|
| Greek categories | Architectural elements |
| Quantity<br>Activity<br>Quality | Form., Pattern, Structure, Geometry, etc.<br>Function, Needs, Effects, Exchange, etc.<br>Meaning, Association, Resemblance, Style, etc. |
| **Secondary Categories** | |
| Substance<br>Relation<br>Will | Construction, Materials, Design, etc.<br>Context, Community, Nature, Feeling, etc.<br>Spirit, Power, Politics, Attitudes, etc. |

Capon (1999a) examines Aristotle's categories of good in architecture by aligning them with ancient Greek virtues and professional practice values. He identifies two main categories: Primary and Secondary. The primary category aligns Greek virtues such as Justice, Temperance, and Wisdom with professional values like Impartiality, Efficiency, and Integrity. The secondary category connects virtues like Duty, Love, and Courage with values such as Responsibility, Respect, and Motivation.

Capon builds on these models by incorporating Aristotle's principles of good performance into six architectural principles, categorized into primary and secondary elements. Primary principles include:

1. Impartiality of Form: Objectivity in form.
2. Efficiency of Function: Efficiency and economy in function.
3. Integrity of Meaning: Propriety and integrity in meaning.

Secondary principles are:

4. Obligations of Construction: Responsibility in design and construction.
5. Regard for Context: Sympathy for context and community.
6. Motivation of Spirit: Motivation and conviction in will and spirit.

In the second volume of his book, Capon examines definitions of architecture provided by several 20th-century architects across different years and texts. He endeavors to categorize their perspectives under the three Vitruvian categories, as outlined in Capon (1999a). Mahmoodi, 2001 compiled the definitions as in table 1-4:

*Table 1-4 Definition of 20· century architects of architecture. These terms are compared with the original Vitruvian categories (Introduced by Capon, 1999b, pp. 349-353, compiled by Mahmoodi, 2001, p. 55)*

| **Vitruvius,** + 2OOO years ago | ***Firmittas* (Firmness)** | ***Utilitas* (Commodity)** | ***Venstustas* (Delight)** |
|---|---|---|---|
| Geoffrey Scott, 1914 | Construction | Convenience | Aesthetics |
| Auguste Perret, 1923 | Material | Use | Beauty |
| Le Corbusier, 1923 | Construction | Utilitarian needs | Custom/tradition |
| Le Corbusier, 1923 | Construction | Needs | Mathematics/harmony |
| Le Corbusier, 1923 | Constructing | Living | Conceiving |
| Le Corbusier, 1923 | Economy | Sociology | Aesthetics |

| | | | |
|---|---|---|---|
| Walter Gropius, 1924 | Technology | Economy | Form |
| Walter Gropius, 1924 | Construction | Economy | Design |
| Walter Gropius, 1924 | Technical | Economic | Aesthetic |
| Walter Gropius, 1924 | Technical | Social | Aesthetic |
| Walter Gropius, 1924 | Structure | Function | Intellect |
| Ludwig Mies van der Rohe, 1928 | Technical | Economic | Cultural |
| Ludwig Mies van der Rohe, 1928 | Material | Functional | Spiritual |
| Ludwig Mies van der Rohe | Technical | Economic | Architectural |
| ASNOVA, 1931 | Technical plausibility | Economic feasibility | Plastic expression |
| Nikolaus Pevsner, 1943 | Construction | Function | Style |
| Reyner Ban bam, 1960 | Structural | Social | Academic |
| L. Benevolo, 1960 | Technical | Social | Cultural |
| Christian Norberg-Schulz, (1963) | Technical | Functional | Aesthetic |
| Christian Norberg-Schulz | Physical | Social | Cultural |
| Christian Norberg-Schulz | Techniques | Building task | Form/semantics |
| Robert Venturi, 1966 | Structure | Programme | Expression |
| N. L. Park, 1968 | Construction | Function | Aesthetics |
| N. L. Park, 1968 | Physical | Behavioural | Conceptual |
| George Baird, 1969 | Technique | Function | Form |

| | | | |
|---|---|---|---|
| Charles Jenks, 1969 | Technics | Function | Form |
| L Ligo, 1974 | Technics | Function | Form |
| David Canter, 1977 | Physical attributes | Actions | Conceptions |
| R. Krier, 1982 | Construction | Function | Form |
| M. Foster, 1983 | Structure | Design | Style |

The prevailing definitions concerning the three Vitruvian categories are as follows: Firmness aligns most closely with aspects like construction and technique; Commodity relates primarily to economic and social considerations; and Delight pertains to the formal and aesthetic aspects of architecture. The only concern in Capon's studies was referencing to Vitruvius': Firmitas (firmness), Utilitas (commodity), and Vensutas (delight) because the three elements are seen as complementary to each other rather than separable. If delight is separated from commodity, it could imply that delight serves no fundamental purpose. In addition, firmness and commodity are considered main contributors to delight. And in this sense, architectural design thinking is involved with problems that are related to those three cores. Although, the three cores are redescribed from time to time, they remain the main cores to assess an architectural design either as good one or not from a general perspective.

Lawson (1990) argues that architectural design defies strict boundaries, involving subjective value judgments and a blend of problem discovery and resolution. Mahmoodi (2001) suggests that despite its intuitive nature, design approaches can be categorized by types. Broadbent's (1988) comprehensive categorization includes Pragmatic, Typologic, Analogic, and Syntactic Design. Pragmatic Design employs trial-and-error processes based on physical factors, while Typologic (iconic) Design utilizes pre-established solutions. Analogic Design draws inspiration from various sources to foster new insights, while Syntactic Design (canonic) operates within rule-based systems, often geometric. Mahmoodi notes that architects often combine multiple design types within a project to address unique challenges effectively.

## 1.3 The black box vs the glass box vs the grey box metaphors

Creative activities within the design process unfold organically, reflecting its iterative and dynamic nature (Lawson, 1993; Lang, 1987; Broadbent, 1969). Designers engage in continuous exploration, synthesizing diverse information, making intuitive leaps, and forging unexpected connections to generate innovative solutions. Bruno Latour's metaphorical concept of the black box, introduced in 1987, symbolizes encapsulated knowledge known to specialists but often perceived as unknowable (Latour, 1987). Black boxes serve as ready-made solutions, offering practical outcomes without necessitating exhaustive internal understanding (Witt, 2018).

Norbert Wiener's dichotomy of black and glass boxes underscores the trade-off between usability and comprehensive understanding in design operations. The "black box" approach, as elucidated by Lawson (1993) and echoed by Lang (1987) and Broadbent (1969), conceptualizes design as an abstract and internalized mental activity. This perspective highlights design's multifaceted and subjective nature, encouraging the use of techniques like brainstorming and synectics to illuminate the creative process (Lawson, 1993). Within this framework, the design process is seen as a complex interplay between analytical and holistic thinking, involving the integration of problem models with visual-spatial and symbolically coded solution patterns (Tovey, 1984).The "black box" approach to design refers to viewing the creative and mysterious aspects of the design process as an abstract and internalized mental activity within the designer. In this perspective, design is seen as a complex and subjective process that defies easy analysis. While traditional analytical methods may struggle to dissect the intricacies of design thinking, techniques like brainstorming and the application of synectics are considered helpful in providing a glimpse into the visualization and ideation aspects of the design process.

Throughout history, designers have fluctuated between black boxing and glass boxing in their relationship with mathematics, representing informed action (glassboxing) and pragmatic approaches (blackboxing) (Witt, 2018). The "glass box" (glass box) design method, emphasized by Broadbent (1969) and Archer (1969), analyzes the design process systematically, considering identifiable events such as identification, analysis, synthesis, and evaluation. This approach utilizes various methods like systems analysis and set theory, providing a logical framework for problem-solving. Figure 1-2 shows the design process as proposed by Archer, 1969.

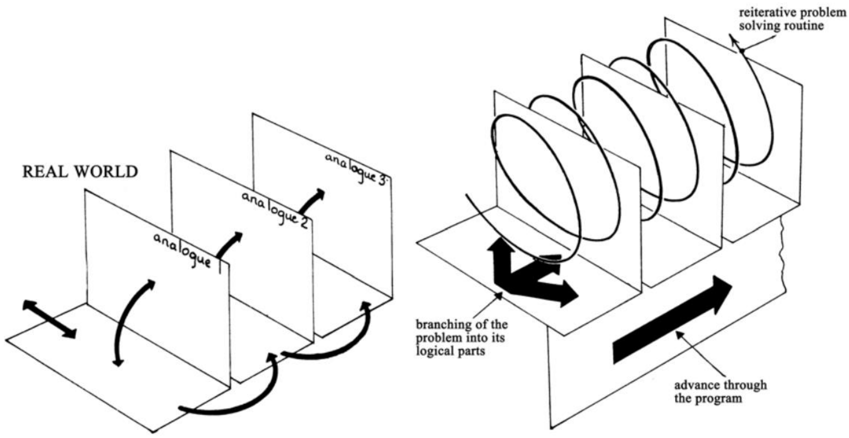



*Figure 1-2 Design process (Archer, 1969, p.94 and p.100)*

The "glass box" design approach was criticized by Sullivan and Hillier (1972) for prioritizing artistic development over building quality. In response, the evolution of design methods integrates concepts from Popper's philosophy of science, emphasizing conjecture and refutation to bridge the gap between science and art in the design process.

In 2018, Witt., A., wrote in Log Journal vol 43 introducing a new term called ‘greyboxing’. Witt introduced the concept of "greyboxing," highlighting its significance in contemporary software development, particularly in the integration of mathematical techniques into digital design. Grey boxing involves orchestrating combinations of black and glass boxes to create novel functions, offering a unique perspective on how mathematical ideas infiltrate architecture. This approach, reminiscent of architecture's historical assimilation of mathematical techniques, facilitates the creation of new operational networks by opportunistically appropriating instrumental knowledge without requiring exhaustive technical understanding.

In the current digital landscape, the use of neural networks exemplifies the gray boxing approach, where the network extracts formal rules from a diverse collection of images to generate novel architectural forms. The process involves navigating through layers of black boxes, emphasizing the importance of curated training sets as a common currency for distilling formal rules. Witt positions these techniques within the broader context of black boxing, gray boxing, and glass boxing approaches in design history, underscoring their enduring relevance and adaptability across disciplines. In general, Witt's description paints grayboxing as a resilient and effective strategy, allowing for creative adaptation and integration of mathematical concepts, computer science, AI., and many other fields into the design process.

## 1.4 Design methodologies

Mahmoodi (2001) distinguishes between 'design methods' and 'design methodology', where the former refers to techniques and procedures, while the latter encompasses the broader strategy and process of applying these methods within the realms of analysis, synthesis, and evaluation. Architecture history traces back the concept of architectural designing to Vitruvius, with later contributions from Alberti (1485), Descartes (1637), Laugier (1753), and Le Corbusier (1923), each emphasizing aspects of problem-solving, decomposition, and composition in the design process. Contemporary architects and critics view the design process as a blend of reason and intuition, described as "learning-by-doing" (Grant, 1975, 1982; Schon, 1984). Today, the focus is on incorporating user behavior and enhancing the built environment for sustainability, with Mahmoodi identifying two major models of design methodologies: 'the systematic model' and 'the environmental model'.

***The systematic model***

Throughout history, societal shifts have significantly influenced the design process. In the Middle Ages, architecture revolved around religious buildings, while today, civic and business structures dominate urban landscapes. Post-World Wars, Modernism emerged, emphasizing simplicity and speed. Mahmoodi (2001) introduced the "stage-phase approach" in design methodology, influenced by Green (1962), Cutler and Cutler (1982), and others. Broadbent (1973) highlighted engineering's impact on design theory, while Asimow (1962) outlined problem-solving phases. Gugelot (1963) introduced a six-stage method for design education. The AIA's model (Duerk, 1993) includes Pre-Design, Schematic Design, and other steps. Salvadori (1974) divided architectural praxis into Programming, Schematic, and other phases. Duerk (1993) emphasized interactive processes between Analysis, Synthesis, and Evaluation. Figure 1-3 shows the proposed design process by Duerk (1993).

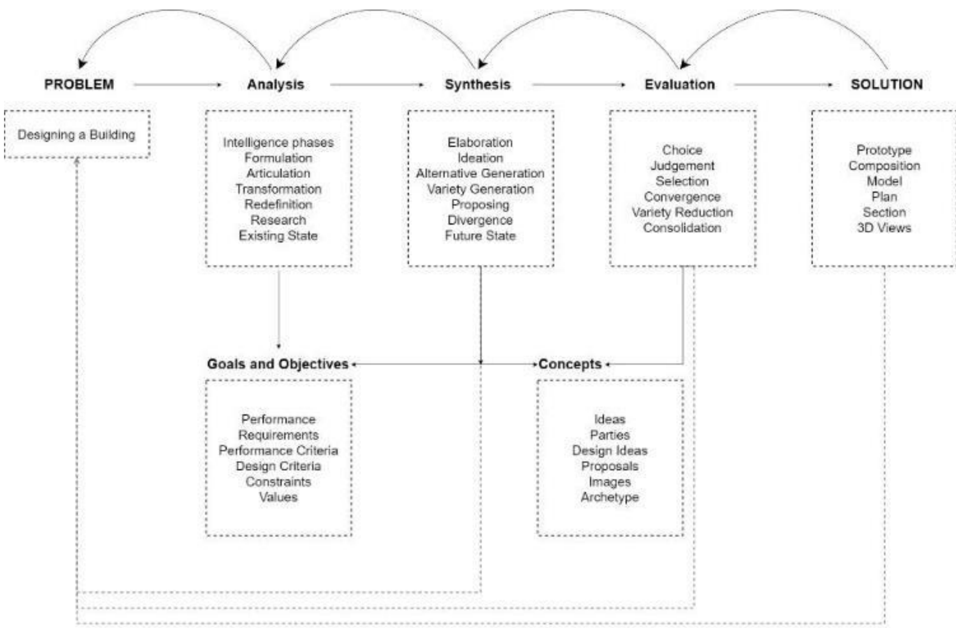



*Figure 1-3 The Design Process: Analysis, Synthesis, and Evaluation (Duerk 1993, p. 18)*

As seen from the figure, Duerk's model suggests cyclical relation between activities with no order of priorities. In addition, the model separates the activities each set in its own category where this does not meet what happens in practice where many activities are inseparable.

Advancements in technology with the introduction of Building Information Modeling (BIM) has reshaped the design process in the AEC industry. BIM integrates inputs from all disciplines involved in a project, facilitating synthesis, analysis, and evaluation in a unified framework. Optimization techniques are applied to various design aspects, such as views, structure, daylighting, and ventilation. This integrated approach emphasizes the importance of rational decision-making in the design process.

In the 1960s, early environmental design models leaned towards discrete, sequential decision processes influenced by "rational" decision-making models (Simon, 1957, 1960, 1969). Newell, Shaw, and Simon's 1957 paper introduced the information processing theory, emphasizing cognitive processes (Rowe, 1987). Lang (1987) highlights Studer's (1970) model, focusing on defining function, designing form, building, and evaluating. Computer-aided design approaches, like ICADS, emerged, emphasizing collaborative decision-support systems. Mahmoodi (2001) criticizes overreliance on computer systems in design, advocating for human involvement in decision-making.

***The environmental model***

According to Mahmoodi (2001), the 'environmental model' in design methodology incorporates environmental considerations and human sciences. Lang (1987) advocates for an argumentative design process, delineating phases like Intelligence, Design, Choice, Implementation, and Post-implementation Evaluation. This mirrors professional praxis, which includes Programming, Design, Evaluation and Decision, Construction, and Post-occupancy Evaluation (Lang, 1987). Intelligence activities in design involve problem identification and understanding, goal formulation, and environmental evaluation (Lang, 1987). Mahmoodi (2021) supports the environmental approach, stressing the context-dependent nature of problem perceptions and definitions. However, criticisms include the perceived linearity of the model, even in its interactive form (Mahmoodi, 2001). Another model by Professor Broadbent (1988) incorporates Popper's (1963) "conjecture" and "refutations." The model explores different design types (conjectures), including Pragmatic, Typologic, Analogic, and Syntactic Design. It also examines the fit of spaces to activities, environmental filtering, cultural symbolism, and environmental impact (refutations) (figure 1-4).

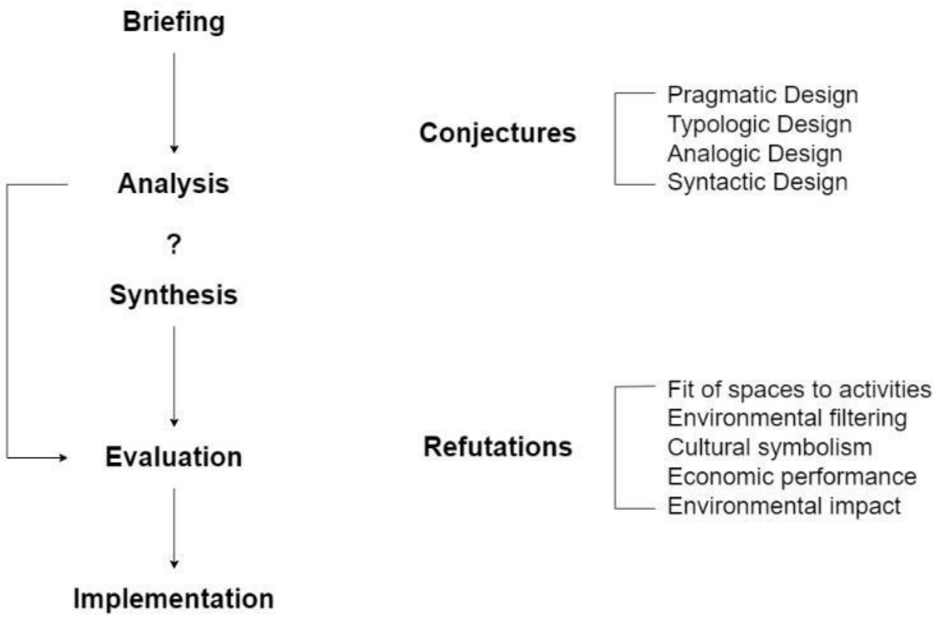



*Figure 1-4 The environmental model of the design process (Broadbent, 1988, p. 467)*

Professor Broadbent (1988, p.459) clarifies that his model of the design process deviates from so-called "linear" models. Instead, he describes it as a "map" of the "design territory." Later in the same text, he emphasizes that the design process is not confined to a linear sequence and can commence at any point within his model.

Broadbent's latest proposal reflects the contemporary architectural design process, involving iterative analysis and evaluation after each design iteration. Stakeholders assess various aspects like environmental, social, aesthetic, functional, cultural, and economic factors until the building's performance is reviewed and decisions are made.

## 1.5 Architectural design process

Design process methodology, established as a distinct discipline in the 1960s, has evolved beyond traditional approaches like 'drawings and patterns.' Although these conventional methods persist, the field has embraced more diverse and advanced techniques.

As design methodology gained recognition as an independent field, scholars like J. Christopher Jones, in his 1980 book "Design Methods: Seeds of Human Future," highlighted the intricacies of the design process. Jones identified three key operations: gathering information, testing design decisions, and evaluating their appropriateness. He also categorized the design process into unconscious and conscious intellectual activities, tapping into both intuitive and rational thinking.

Architectural design profoundly influences project outcomes, shaping functionality, aesthetics, user experience, and sustainability. Decisions made during design impact budget, regulations, and adaptability. Collaborative approaches are crucial for success.

Design, as discussed by Chandrasegaran, Kisselburgh, & Ramani (2012), is an iterative process aiming to achieve functionality through form proposal and analysis. However, in the contemporary context, design expands beyond functionality, integrating intellect and technicalities (Royal College of Art, 1979; Cross, 2006). Architects are challenged to infuse creativity into projects, enhancing user experiences while meeting functional needs (Levin, 1966).

Contemporary design thinking diverges from traditional linear methodologies (Abowardah, 2016). Rather than following a strict sequence of analysis, synthesis, and evaluation, modern approaches allow for flexible engagement of different thinking modes throughout the process. This adaptability acknowledges the complexity of design activities. Abowardah (2016) explored practical design methodologies by analyzing insights from renowned architects like Frank Gehry, Zaha Hadid, Toyo Ito, and Peter Zumthor. Gehry's process, for example, involves sketching continuously, drawing inspiration from sculptors and painters. Despite appearing sequential, his method incorporates trial and error, resembling Broadbent's cyclic approach between analysis and evaluation. Figure1-5 shows Frank Gehry’s sketch of Guggenheim Bilbao against the constructed version.

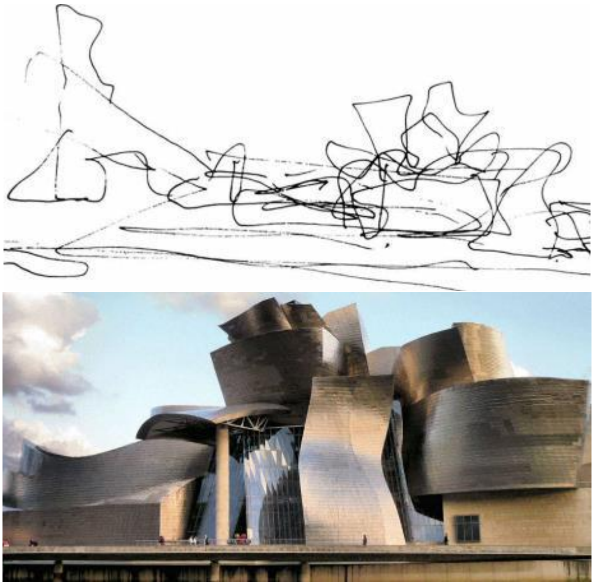


*Figure 1-5 - Frank Gehry's Guggenheim Bilbao - https://architizer.com/blog/practice/tools/how-architecture-is-born-frank-gehry/*

Zaha Hadid Architects utilize digital techniques, known as 'form finding', relying on mathematical studies and optimization for structure, daylighting, and ventilation. Their process remains within the analysis-evaluation cycle, prioritizing digital methods before synthesis. Additionally, Toyo Ito's approach is similar, focusing on integrating with nature through digital tools like Voronoi, aiming to break away from homogeneity in cityscapes. Moreover, Peter Zumthor follows a comparable process but emphasizes space and materials in the analysis phase, relying on physical models for concept evaluation.

Abowardah (2016) outlines the design process into three phases: analyzing knowledge, developing tools, and invention, aligning with Broadbent's suggested approach influenced by Popper's thoughts. Architectural firms globally invest in research and development to automate processes and enhance efficiency.

## 1.6 New technologies effects on the design process

New technologies have revolutionized the design process, particularly in the analysis and evaluation stages. Instead of relying solely on sketching or modeling, advanced design techniques driven by developments in materials and construction allow for more innovative solutions. Architects now trust computers to find optimal forms based on various considerations like environmental impact, spatial relations, and structural analysis. However, architects still direct these approaches based on project goals, ensuring that the final design meets objectives and can be feasibly constructed. Figure 1-6 shows how decisions could be made and buildings forms could be optimized based on solar radiation simulations.

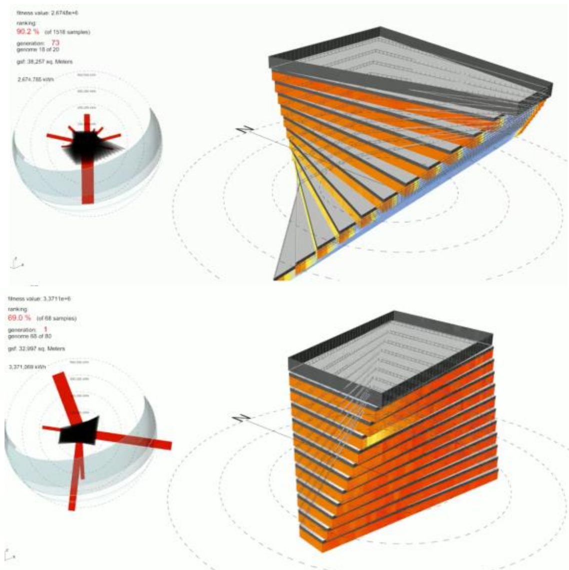


*Figure 1-6- Solar Radiation Optimization with Grasshopper / Galapagos / DIVA by Yazdani Studio, https://yazdanistudioresearch.wordpress.com/2015/02/09/building-optimization-tools-the-grasshopper-definition-and-breakdown/*

After this result, the product is refuted and evaluated against other design problems and aspects until a final decision is made.

Moreover, the approach to deal with the building as information which are sometimes complicated led to changes in the implementation phase where the required product of the process is no longer 2D drawings, but a building information model containing every single element to be constructed and showing the relations and quantities of the elements.

In addition, the development in digital fabrication specialization led to a groundbreaking leap in the preparation of the construction drawings which made building complex geometries very easy and well organized in site. Figure 1-7 shows the use of Grasshopper for Rhinoceros3D in digital fabrication, numbering the surfaces, and extracting all of the information needed for site fabrication. Different plugins are developed to automate the extraction of such information.

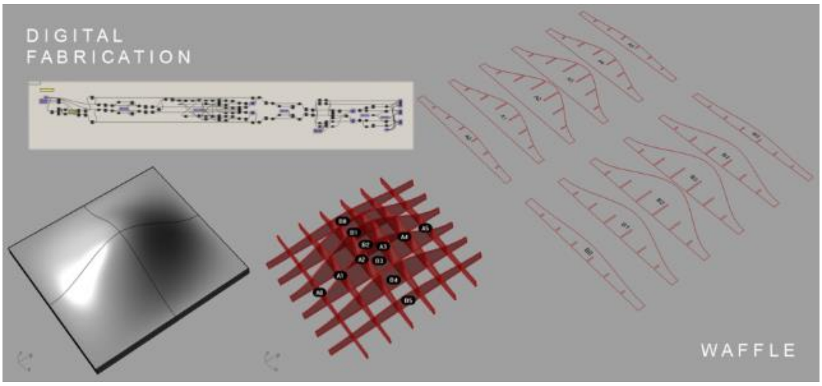



*Figure 1-7- Digital Fabrication Technique to Extract Information about Elements Using Generation, Grasshopper, and Rhinoceros3d – by Antonio Turiello, https://www.grasshopper3d.com/group/generation*

Other techniques such as procedural modeling, simulations, kinetic architecture, and more, also led to many changes in the design process. These techniques will be discussed in chapter 2.

To conclude, recent technologies did not affect the design process suggested by Broadbent, 1988, per say, but it affected the phases in a very strong and efficient way that made the cyclic process between analysis and evaluation much smoother and made the intervention of different design types and modes applicable at any stage of the cyclic process.

## 1.7 Complexities in architectural design

Since Vitruvius, architectural evaluation has revolved around firmness, commodity, and delight. Attempts to quantify delight, such as through the golden ratio or modular grids, have persisted throughout history. Today, environmental parameters can be quantified using tools and building codes, enabling assessments of factors like daylighting and power consumption.

However, assessing functionality faces challenges due to the uniqueness of each design. Support tools are often used late in the process, hindering early identification of imperfections. Post-occupancy evaluations focus on design confirmation rather than comprehensive quality quantification (Mahmoodi, 2001).

To improve design quality assessment, there's a need for research into tools that accommodate design uniqueness and can be applied in the early phases. Bridging the gap between support tools and design requirements is crucial. Without such advancements, aesthetic assessment will remain subjective, relying solely on designers' perspectives of beauty and proportions.

It is crucial to understand the difference between problem solving and designing. In problem-solving, designers typically seek logical solutions to specific issues, whereas in designing, they create a comprehensive solution that addresses multiple design problems. Designing involves more effort, as designers need to integrate both creative and logical solutions. Despite the complexity of design, the notion of breaking down design problems into constituent parts for easier solutions is challenging, as Professor Lawson notes (1993), emphasizing the need to consider the entire problem or a multitude of issues simultaneously. Architects can face many challenges in design categorized by the author as: complexities within thinking, complexities in problems definition, complexities within the design process, complexities in decision making, and complexities resulting from rapid development in technology.

***Complexities in Problems Definition***

Architectural design involves solving complex problems, from client requirements to environmental impacts. These intertwined issues create a network of decisions for architects. Defining problems accurately is challenging due to many non-quantifiable aspects of architecture, leading to uncertainties. Lateral thinking is essential, as solutions may emerge unexpectedly, but delving deeper can reveal further complexities, perpetuating a cycle of problem-solving.

### *Complexities within Thinking*

Design thinking involves multifaceted problem-solving influenced by cognitive styles like divergent vs. convergent and impulsive vs. reflective thinking. Architects balance these styles to tackle design challenges effectively, leveraging different modes of thinking within the same task to innovate.

### *Complexities within the Process*

The design process encompasses analysis, synthesis, and evaluation, involving data collection, idea generation, and solution selection. It's recognized as a series of internalized operations, with designers transforming inputs into outputs, raising questions about the intuitive process governing human thinking in design.

### *Complexities in Decision-Making*

Architectural design requires intricate decision-making due to varied and interconnected problems. Conflicting decisions can arise, leading to complex trade-offs. Some problems require creative problem-solving, while others rely on design principles and standards, though these may have shortcomings.

### *Complexities Resulting from Rapid Technological Development*

Technological advancements, like BIM and AI, have revolutionized architectural design, offering new opportunities and challenges. BIM facilitates information integration, enhancing efficiency, while AI enables tasks like visualization and parameter prediction. These developments prompt questions about architects' roles and idea authenticity in the face of evolving technology.

To conclude, architectural design is a very complex process because of the many complexities encompassed in this activity associated with many aspects like aesthetics, information, technology, constraints, and many others that hinder the problem-solving process. Those complexities need to be understood well to map how an architect solves the problems resulting from them.

**Summary**

This chapter delves into architectural design thinking, exploring its complexities and evolution. It begins by examining the human brain's cognitive processes, drawing from psychological studies on problem-solving from Associationism to Cognitivism, highlighting the relevance of Cognitivism in architectural thinking due to its emphasis on information processing. Definitions of architecture and design types by Vitruvius and Broadbent are discussed, revealing the challenge of forming a concrete definition.

The exploration extends to understanding the architect's thought process through black box and glass box metaphors, emphasizing the complexity and non-linear nature of architectural design. Various complexities within thinking, problem definition, design process, decision making, and technology are examined, showcasing the multifaceted nature of architectural design.

The chapter transitions to the Architectural Design Process, introducing J. Christopher Jones' operations and tracing the evolution of design methodologies since the 1960s. Mahmoodi's classification into systematic and environmental models is discussed, along with specific models like Broadbent's stage-phase model and Archer's design process flow chart.

Insights from renowned architects like Frank Gehry, Zaha Hadid, Toyo Ito, and Peter Zumthor are incorporated, showcasing diverse design approaches. Abowardah's investigation of practical models for design methodology, analyzing famous architects' processes, is explored.

The impact of new technologies on the design process is examined, highlighting advancements like solar radiation optimization tools and Building Information Models (BIM), which have streamlined decision-making and construction processes. The chapter concludes with a discussion on evaluating architectural design outcomes, emphasizing the need for early assessment to avoid problems. Parameters affecting form-making, known as delight, are discussed, acknowledging the challenge of assessing aesthetics due to changing principles throughout history.

In summary, the chapter provides a comprehensive overview of architectural design thinking, from cognitive processes to practical methodologies and the influence of technology, emphasizing the dynamic and creative essence of design in architecture.

Yet, the question remains, can the process occurring in the architect's mind within a black box approach be mapped? Or can the architect themselves map the process of thinking? Does the process remain the same for the architect when a black box approach is taken regardless of the project? Answering these questions could have a great impact on the architectural design process. Especially if the answer is true, then supposedly a machine can learn this process and act as a designer.

# Chapter 2: Mapping the Elements of Forms in Architecture

**Preface**

Architectural form extends beyond the arrangement of physical space and activities. According to Yilmaz (1999), it serves as a meaningful instrument, signifying a connection with elements, their syntax, associated meanings, and their impact on individuals. The reduction of form to the mere selection and organization of elements is cautioned against; it should not be seen solely as a tool for conveying meaning.

The foundational elements - point, line, plane, basic shapes, and solids - historically influenced various conceptual geometries, continuing to play a crucial role in contemporary architecture. These elements significantly shape space and architectural form, contributing aesthetic, symbolic, and conceptual depth to architectural design. Architects widely employ these elements, particularly in the organization of architectural space.

When the use of elements is symbolic and serves as surface decoration, the elements are usually referred to as motifs (Aamir, 2017).

In this chapter, architectural form is defined beside mentioning different characteristics of form as well as its elements from points to solids. After that, the difference between form making and form finding is highlighted as a base for the upcoming question in next chapters whether AI could serve in the form making or form finding. The chapter ends by discussing how to map different contemporary architectural form elements and motifs. This part will later be serving in translating those elements to computational parameters that will be 'learned' by AI algorithms.

## 2.1 Characteristics of architectural forms

Erzen (2015) posits that in ancient times, the Greek 'idea' equated to the Latin 'form,' suggesting that form originates from an underlying idea or purpose, making them inseparable. Bacon (1974) defines architectural form as the intersection of mass and space, comprising elements like texture, light modulation, and color, which collectively imbue space with a distinct quality. Ching (1996) discusses the versatility of the term "form," encompassing external appearance, conditions of presentation, and formal structure in art and design. In "Form, Space, and Order," Ching presents form as both internal structure and external outline, emphasizing unity. Forms have relational properties governing composition and arrangement, including position, orientation, and visual inertia.

In architecture, forms are categorized as regular or irregular. Regular forms, as per Ching (1996), typically exhibit stability and symmetry around one or more axes. Even after dimensional changes or the addition/subtraction of elements, their inherent regularity often remains. Irregular forms, on the other hand, are asymmetrical and dynamic, accommodating both solid masses and spatial voids. The spatial organization of architectural elements, influenced by these forms, defines compositions' visual dynamics and overall design (Ching, 1996).

Architectural form theories have evolved since ancient times, notably influenced by Plato's multifaceted concept of form. The Renaissance marked a shift towards viewing the Idea as originating in the artist's mind rather than a unified essence. In the seventeenth century, Boullee emphasized the epistemological meaning of form, prioritizing geometric designs for instant perception. By the twentieth century, the Gestalt psychology of form introduced by figures like Le Corbusier highlighted the unity of perception and conception. However, there was a departure from mental realms of form in favor of methodological approaches, with figures like Alexander proposing mathematically based design methodologies. Eisenman advocated for eliminating preconceived forms, favoring generative grammar in the design process. The advent of computers further transformed form, making it an expression of universality and methodological rather than purely aesthetic or epistemological (Plato, Renaissance, Boullee, Le Corbusier, Alexander, Eisenman).

***Geometry and form***

From Vitruvius' geometric ideals to modern approaches like Corbusian regulating lines and Miesian modular grids, architectural design has relied on mathematics (Burry, 2010). In ancient times, even the angle of inclination of the Great Pyramid was determined through geometric constructions (Yilmaz, 2016). During the Middle Ages and the Renaissance, mathematical principles were considered divine and formed the basis for designing beautiful buildings. Greek, Roman, and Renaissance architects grounded their aesthetic rules in geometric ratios, with geometry serving as the primary mathematical tool until the seventeenth century. As buildings grew in complexity in the late nineteenth century, a science of structural design emerged, employing sophisticated geometric operations. Throughout history, the prevailing notion has been that architectural form must adhere to mathematical principles, establishing order through regularities, proportional systems, and synthetic methods for generating forms. Architecture has aimed to emulate the geometric order of nature, offering schemes for analyzing finished forms.

Yilmaz (2016) highlights how ancient Greek and Egyptian architects used geometry and proportion to seek divine rules for form generation. They established foundational units and precise geometrical systems, deriving proportions from geometric figures to create clear and rational designs with symmetry. Geometry represented divine truths in ancient Egypt, influencing timeless design principles in Western theory. Pythagoras influenced the distinction between matter and form, while Plato's theories on ideal forms, based on Pythagorean ratios and Platonic solids, shaped architectural design principles. (Figure 2-1).

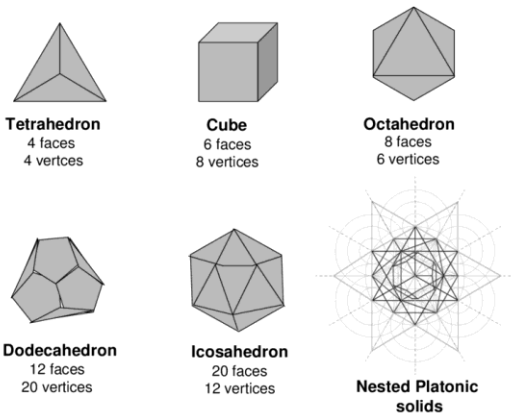



*Figure 2-1 - Platonic Solids, Hill, V., and Rowlands, P., 2008, Nature's Code*

Architects later adapted Platonic Solids into practical building blocks like the sphere, cylinder, cone, pyramid, and cube, reflecting a pragmatic approach to architectural form.

Medieval architects used basic geometric shapes like circles, equilateral triangles, and squares to create intricate forms in both section and plan, embodying divine characteristics (Yielmaz, 2016). The Gothic cathedrals employed two design schemes, Ad Quadratum and Ad Triangulum, for proportioning building plans and facades, determining element sizes, and creating repetitive ornamentation. (figure 2-2).

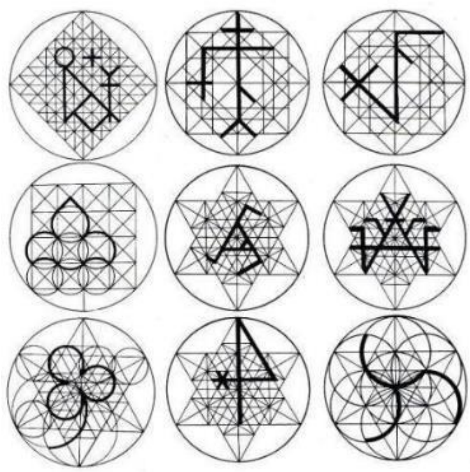


*Figure 2-2 - geometrical analyses of mason's marks on differentdrawings of Gothic cathedrals, Franz von Rhiza, Studien über Steinmertz Zeichen,1917, pp. 44-45*

These geometrical principles were foundational in medieval building design long before they were systematically codified during the Renaissance.

During the Renaissance, architects like Alberti and Bramante applied idealized geometric concepts influenced by Greek mathematical systems, as outlined by Yilmaz, 2016. They viewed architecture as mathematics translated into spatial units, applying Pythagorean ratios to create proportional harmony in buildings. This period saw experimentation with ideal forms and proportions, exemplified by Leonardo's Vitruvian Man, reflecting a belief in buildings belonging to a higher order governed by universal truths.

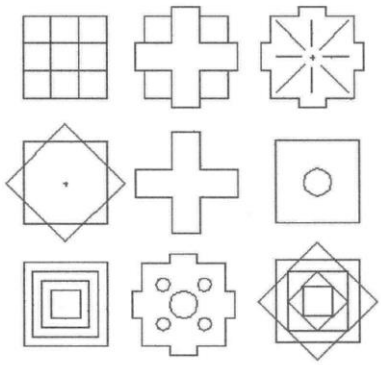


*Figure 2-3 - Diagrams of St. Peter Church by Bramante, Yilmaz, 2016, Evolution of the Architectural Form Based on the Geometrical Concepts*

In Baroque architecture, Guarini utilized dynamic geometrical operations to create complex spatial designs, departing from Renaissance clarity (Yilmaz, 2016). During the Enlightenment, architects like Boullee and Ledoux employed ideal geometry symbolically, while Durand advocated for simplicity and economy (Yilmaz, 2016). In the twentieth century, architects such as Le Corbusier and Mies Van der Rohe embraced Euclidean geometry, emphasizing rationality and integrity in design (Yilmaz, 2016). A minority, including Frank Lloyd Wright, blended geometric concepts with experimental psychology to design space logically, integrating geometry, volumes, and functions harmoniously based on geometric principles.

In general, architectural forms and spaces are composed of four fundamental element types: points, lines, planes, and volumes. In the context of architecture, these elements are typically three-dimensional volumes defined by vertices (points), edges (lines), and surfaces (planes). Each element type holds intrinsic characteristics within an architectural context. Lines convey direction, emphasizing significant end points and defining boundaries. The intersection of lines introduces a third point, providing additional content for making relative judgments of distance and angle. In architecture, a three-dimensional design can be envisioned mentally before it materializes physically.

The perception of three-dimensional forms can vary based on factors such as viewing angles, distances, lighting conditions, color, and texture. However, certain elements can be considered independent of these variable situations including shape, texture, light, color, size, and scale (Yilmaz, 2016). Forms could be significantly manipulated by changing these elements. However, shapes and sizes remain the most relating aspects to how a form is perceived.

Additionally, Yilmaz (2016) emphasizes that design strategies, such as unity, balance, contrast, harmony, rhythm, and proportion, play a crucial role in shaping the overall concept of a "whole" throughout the design process, whether in two-dimensional or three-dimensional contexts. Today, parametric design empowers architects to craft visually appealing façade patterns, seamlessly integrating complex surfaces with digital fabrication techniques. These patterns, enhanced by strategies like repetition, rhythm, harmony, and unity, often serve as the focal point, requiring minimal manipulation of the form itself.

## 2.2 Form generation in architectural design: form making vs. form finding

The reciprocal relationship between mathematics and information technology (IT) has led to the development of new mathematical tools, particularly in the field of architecture. This evolution goes beyond creating variations of functional solutions, drafting, modeling, and presenting—it extends to the actual generation of forms. This approach is known as the form-generation process. Initially rooted in using rules and algorithms for architectural forming in a one-way process, the form-generation process has evolved to incorporate a comprehensive set of parametric equations. These equations can be recorded and revisited at any stage of the design, allowing for dynamic changes to parameters and subsequently altering the entire design solution swiftly (El Iraqi and El Daly, 2017).

The generation of architectural forms involves a range of techniques and methodologies, often categorized into form-making and form-finding processes. While the form-making process emphasizes the subjective, intuitive, and creative aspects of design, the form-finding process relies on mathematical rules, algorithms, or constraints to guide the generation of architectural forms. The choice between these approaches often depends on the goals of the design, the preferences of the architect, and the intended expression or functionality of the final structure. Many architects may incorporate elements of both processes, striking a balance between creative freedom and systematic control in their design methodologies. In contemporary architectural practices, forms are increasingly designed to align with specific parameters and functions.

***Form making:***

According to El Iraqi and El Daly, 2017, form making is a creative process driven by intuition and imagination, often preceding analysis and design constraints. This direct embodiment of ideas into forms can be facilitated through conventional or computational mediums, sometimes resembling sculpture with a focus on form over function.

In the realm of architectural creativity, diverse theories have profoundly shaped form making. Anchored in principles like intuition, unpredictability, metaphorization, and departure from strict logic, these theories offer architects unique perspectives. Intuition guides architects beyond analytical rigidity, while unpredictability injects spontaneity. Metaphorization relies on symbolic associations, fostering imaginative structures. Rejecting strict logic encourages exploration of unconventional, non-linear approaches, leading to organic and innovative forms. Established theories like Wallas's stages of incubation, and concepts such as Genploration and synectics, further enrich the creative toolkit. Embracing these theories allows architects to synthesize intuition, metaphor, and unpredictability, transcending conventional boundaries in architectural form creation. Some of these theories are mentioned in table 2-1.

*Table 2-1- Analogue Form Making Theories (El Iraqi and El Daly, 2017)*

| Theory | Concept | Role in Form Making |
| --- | --- | --- |
| **Intuition** | Intuition refers to the ability to foresee without necessarily understanding the process. | Architects rely on intuitive processes to generate ideas directly without a rigid analytical approach. |
| **Unpredictability** | Unpredictability emphasizes the unexpected nature of creative processes. | Form making benefits from elements that are not pre-determined, introducing spontaneity and uniqueness. |
| **Metaphorization** | Metaphorization involves using metaphors and associations in the creative process. | Metaphors serve as powerful tools for generating novel and imaginative forms, connecting diverse concepts. |
| **No Logic** | This perspective challenges the notion of strict logical processes in creativity. | Architects explore unconventional and non-linear approaches, allowing for more organic and innovative forms. |
| **Theories of Incubation (Wallas)** | Wallas proposed a theory of creativity involving stages like | The incubation stage allows ideas to develop |

| | | |
|---|---|---|
| | preparation, incubation, illumination, and verification. | subconsciously, contributing to the creative process. |
| **Genploration (Finke, Ward and Smith)** | Genploration focuses on the exploration of generic structures and elements in creativity. | Architects explore generic elements that can be adapted and transformed into unique forms. |
| **Redundant Generation (Lem)** | Redundant generation involves creating variations of a concept until a novel solution emerges. | Architects experiment with multiple iterations, allowing for unexpected and innovative forms. |
| **Synectics (Gordon)** | Synectics emphasizes making connections between seemingly unrelated concepts. | Architects draw inspiration from diverse sources, fostering creative connections and generating novel forms. |

The integration of these theories provides architects with a rich toolkit for form making. By embracing intuition, unpredictability, metaphorization, and alternative logic, architects can push the boundaries of creativity and produce innovative architectural forms.

In recent decades, digital systems have revolutionized architectural design, impacting geometric representation and design synthesis. Methodologies like fuzzy modeling and random functions, highlighted by El Iraqi and El Daly, 2017, offer innovative approaches. Fuzzy modeling introduces imprecise images akin to architectural sketching, while random functions challenge architects to transform computer-generated shapes, rooted in chaos theory. Integration of 3D digitizing allows seamless navigation between real and digital environments, as seen in Gehry's work, blending tangible and virtual realms in architectural design. (figure 2-4).

*Figure 2-4 Working model of the 1989 Vitra Design Museum in Germany by Gehry, Frank O. Gehry, courtesy Frank Gehry Papers at the Getty Research Institute*

### *Form finding*

Form finding in architecture involves discovering and shaping architectural forms exclusively derived from function, employing rules, constraints, and algorithms within a "generator." This process, outlined by El Iraqi and El Daly, 2017, encompasses both analogue and digital methods. Analogue form finding, exemplified by architects like Jean-Nicolas-Louis Durand, utilizes mathematical rules or transformational principles to generate architectural forms (figure 2-5).

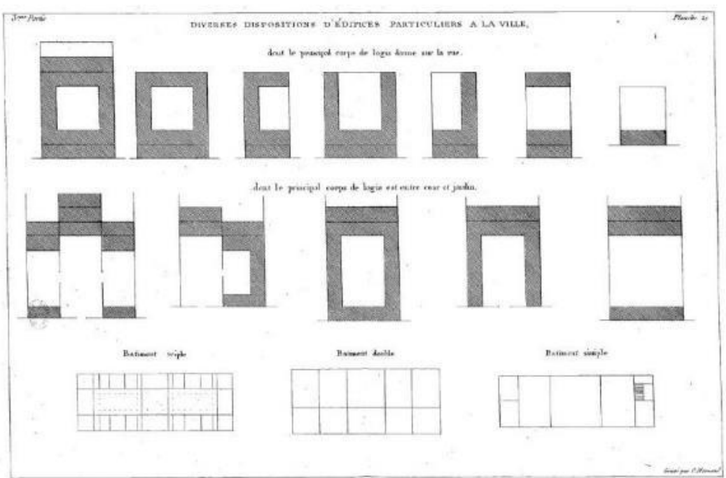



*Figure 2-5 Systemization and Composition by Jean-Nicolas-Louis Durand - https://www.sensesatlas.com/jean-nicolas-louis-durand/*

Similarly, Louis Sullivan employed analogue methods to describe processes for reproducing floral ornamentation based on geometrical constructs (figure 2-6).

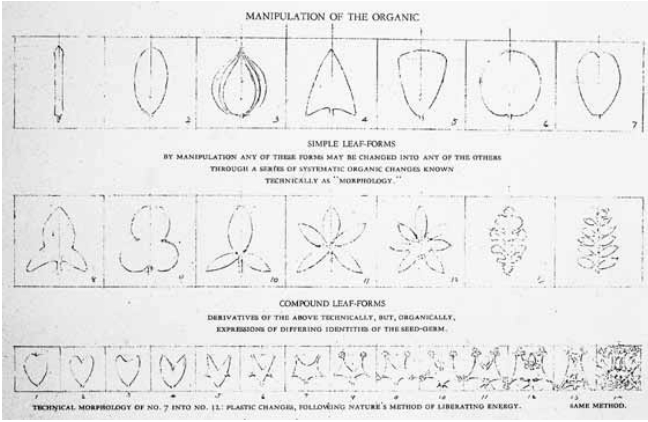



*Figure 2-6 - Louis Sullivan, Plate 2, detail, from A System of Architectural Ornament According with the Philosophy of Man's Power (New York: AIA Press, 1924; reprint 1934).*

Le Corbusier's Five Points of Architecture exemplifies an analogue generative system predating widespread computational use in architecture. Similarly, Peter Eisenman employed analogue transformational rules in design synthesis, creating a system allowing infinite expressions with finite means. Eisenman's approach is evident in his designs of a series of houses as seen in figure 2-7.

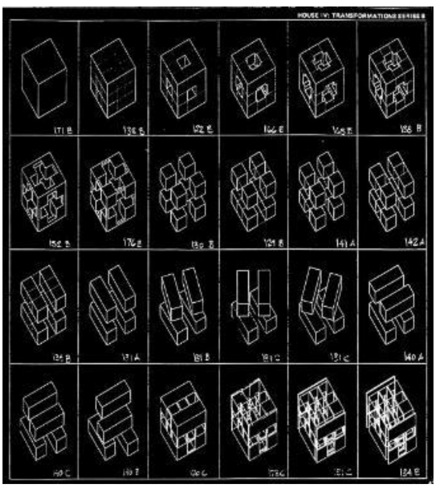


*Figure 2-7 - House IV - Transformations Series by Peter Eisenman - https://eisenmanarchitects.com/House-IV-1971*

Digital form-finding in architecture involves the use of computational tools to generate various forms, ranging from orthogonal conventional shapes using basic shape algebra and formal logic to complex three-dimensional curves and folding surfaces using trigonometric parameterized functions. Computational algorithmic modeling is a process that applies rules and algorithms within a computational medium, but it lacks the potential for changing a rule with direct manipulation applied to the end result. When the process includes a feedback loop allowing for the modification of rules, it is considered a form generation process, whether it's generative or parametric. Digital form-finding types of process including computational modeling, algorithmic modeling, generative modeling, and parametric modeling will be discussed in chapter 3.

## 2.3 Mapping and analyzing the elements and motifs of contemporary forms

In computer science, an architectural style is a set of design rules or conventions that dictate how the elements and relations of a software system should be organized. It represents a family of systems with a shared set of design goals and constraints. Architectural styles define a specific way in which components (such as modules, classes, or objects) and connectors (communication channels, protocols) are arranged to achieve certain architectural qualities like performance, modifiability, or reusability. An architectural pattern, on the other hand, provides a higher-level abstraction compared to architectural styles. It describes a fundamental structural organization schema for software systems, specifying the overall structure of the system and the patterns for the relationships between its components. Architectural patterns go beyond specifying individual elements and connections; they define a set of predefined subsystems, their responsibilities, and also include rules and guidelines for organizing interactions between these subsystems. In summary, architectural styles focus on the manner in which components and connectors are used, while architectural patterns provide a broader, higher-level template for organizing the overall structure of a software system, including predefined subsystems and their relationships. Both concepts are crucial in guiding the design and development of complex software systems, helping architects make informed decisions based on established best practices and design principles (Clement, P., et. al. 2011).

The same definitions can be directly applied to architectural design of buildings. When the difference between the terms ‘architectural pattern’ and ‘architectural style’ is discussed, both terms result in an architectural approach. However, architectural patterns tend to relate to certain problems in a certain context. For instance, an architectural style tends to relate how architectural elements and components are composed focusing on the approach. On the other hand, architectural patterns relate how the resulting architectural approach may solve problems in different contexts including environmental, social, psychological, and economic aspects.

Architectural design motifs are recurring, often symbolic, themes that are used in the design of buildings and structures. These motifs can be derived from various sources, including cultural, historical, religious, or natural influences. They are employed to create a sense of unity, rhythm, and visual interest in the architecture. Motifs relate to the architectural style more than patterns. They mostly tend to symbolize and give meanings or just symbolize how an architect expresses his beliefs of good architecture regarding proportions, aesthetics, and other design strategies.

Examples of Motifs in Architectural Design include the following:

*Architectural Orders:*

Classical architecture introduced the concept of orders, including Doric, Ionic, and Corinthian, each with distinct styles for columns and entablatures. Orders define the arrangement of architectural elements, contributing to the overall style and cohesion of a building or tradition. Figure 2-8 shows how the use of different classical orders as motifs differs according to columns and entablature types and proportions as well decoration elements.

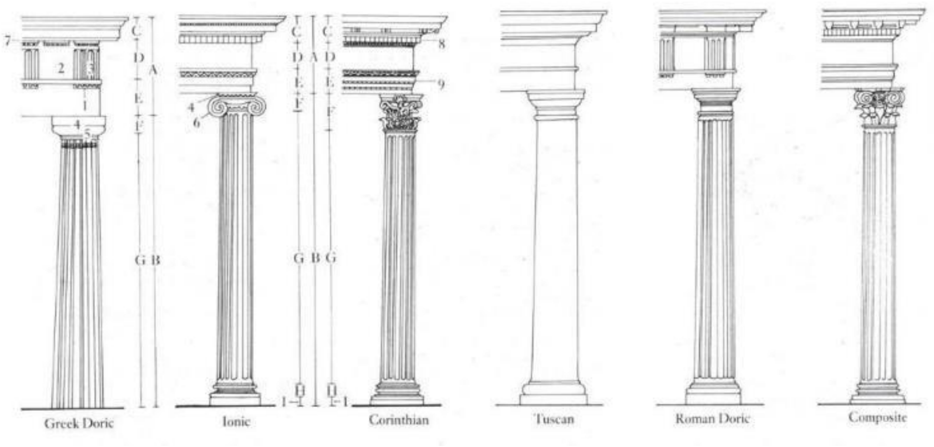



*Figure 2-8 – Architectural Orders from Greek, Roman, and Tuscan Eras,*
*https://blog.stephens.edu/arh101glossary/?glossary=order*

## *Gothic Tracery*

Gothic tracery (figure 2-9), prominent in cathedrals, comprises intricate patterns of intersecting ribs in windows. It enhances aesthetics and establishes a cohesive motif for Gothic architecture through rhythmic recurrence across architectural elements.

*Figure 2-9 – Geometrical bar tracery, Ely Cathedral, Lady Chapel, west window (left), Rayonnant bar tracery above the south rose window in Notre-Dame de Paris (Right) (https://en.wikipedia.org/wiki/Tracery)*

## *Moorish Arch Motif*

The horseshoe arch (figure 2-10) in Islamic architecture, originally from early Christian art during the Roman Empire, is a defining motif characterized by its rounded shape widening at the base. Adorned with intricate patterns and calligraphy, it recurs in doorways, windows, and arcades, contributing to the unique visual language of Islamic architectural styles (Marcias, G., 1954).

*Figure 2-10- Caliphal-style arches of the Taifa palace (11th century) in the Alcazaba of Málaga, Spain, https://en.wikipedia.org/wiki/Horseshoe_arch#cite_note-:02-7*

### *Greek Key Pattern*

The Greek Key pattern, a continuous linear motif in classical design, consists of repeated geometric shapes forming a meandering line. Often used as a decorative border or frieze, its repetitive presence adds rhythm and unity to architectural elements, defining motifs in classical and neoclassical structures. Figure 2-11 (a) shows Greek key on a stove in the in the D.A. Sturdza House, in Bucharest.

### *Art Deco Zigzag Motif*

Art Deco architecture features the distinctive zigzag motif (figure 2-11 (b)), marked by sharp, angular lines. Seen in friezes, cornices, and facade layouts, this pattern represents a departure from traditional ornamentation. Its repetition across elements embodies the dynamic and modern aesthetic of the Art Deco movement in the early to mid-20th century..

*Figure 2-11 –(a) Greek key on a stove in the in the D.A. Sturdza House, in Bucharest, https://en.wikipedia.org/wiki/File:Greek_key_on_a_stove_in_the_in_the_D.A._Sturdza_House,_in_Bucharest.jpg, (b) Zigzag Motif on Smith and Chambers building, Napier, New Zealand. https://edition.cnn.com/style/article/napier-art-deco-architecture/index.html*

*Modernist Grid Motif*

Modernist architecture adopts the grid motif, featuring regular and geometric arrangements of elements like windows and columns. Architects such as Ludwig Mies van Der Rohe, Le Corbusier, and Peter Eisenman apply this motif, inspired by Piet Mondrian's lines. Gerrit Rietveld, influenced by Mondrian, incorporated similar elements into Mrs. Truus Schröder-Schräder's house in Utrecht, Netherlands (figure 2-12). The pervasive repetition of the grid motif reinforces modernist principles of clarity and rationality, becoming a hallmark of the architectural style.

*Figure 2-12- Rietveld Schröder House, https://en.wikipedia.org/wiki/Rietveld_Schr%C3%B6der_House*

Following Le Corbusier and Mies Van Der Rohe, architects like Alvaro Siza, Kenzo Tange, Gordon Bunshaft, Richard Meyer, and Tadao Ando embraced the grid motif. Today, this motif is ubiquitous in contemporary villas and private residential buildings worldwide. Factors such as high construction costs and fast-paced social lives have contributed to its widespread adoption. As a result, designs are becoming increasingly similar, with architects employing a common language, as seen in the analysis of common motifs in Table 2-2.

*Table 2-2 Analysis of Common Contemporary Motifs*

| No. | Project Photo | Description |
|---|---|---|
| **1** |  | - C shaped slab with thickness around 40 cm on the first floor<br>- C shaped terrace elevation on ground floor<br>- The terrace is recessed from the ground floor edge.<br>- Glazing facades inside the slab edges<br>*https://www.flickr.com/photos/aareps/9019441462/in/album-72157636835800726/* |
| **2** |  | - C shaped slab with thickness around 40 cm on the first floor<br>- L shaped terrace elevation on ground floor<br>- Terrace is on the ground floor edge<br>- Glazing facades inside the slab edges<br><br>Private villa in Vilnius, Lithuania by ngarchitects<br><br>*https://ngarchitects.eu/vila-energy/* |

| | | |
|---|---|---|
| 3 |  | - C shaped slab with thickness around 30 cm on the first floor<br>- L shaped terrace elevation on ground floor<br>- The terrace is recessed from the ground floor edge.<br>- Glazing facades and stonework inside the slab edges<br><br>*https://catalog-plans.ru/catalog/62-65* |
| 4 |  | - C shaped slab with thickness around 30 cm on the first floor<br>- L shaped terrace elevation on ground floor<br>- The terrace is on the ground floor edge.<br>- Glazing facades inside the slab edges<br><br>EH House, Pilar, Argentina by Estudio GMARQ. *https://www.archdaily.com/906904/eh-house-estudio-gmarq?ad_medium=gallery* |
| 5 |  | - C shaped slab with thickness around 70 cm on the first floor<br>- L shaped terrace elevation on ground floor<br>- The terrace is recessed from the ground floor edge.<br>- Glazing facades and woodwork inside the slab edges<br>Twelve by Jaime Salvá, Santa Ponça, Mallorca, Spain. *https://homeadore.com/2020/07/27/twelve-by-jaime-* |

| | | |
|---|---|---|
| | | *salva/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed:+homeadore+(HomeAdore)* |

Other motifs appear strongly in contemporary villas and residential buildings, including the use of repeated solid blocks with light or opening gaps (figure 2-13), the use of stripped louvers more as an architectural pattern (figure 2-14), and the use of skewed recessed blocks which usually frame openings (figure 2-15). Even some of these motifs are composed together to form a different look.

*Figure 2-13- Using solid strong walls in contemporary designs (villa by Rymar Studio (left), 21 Villa by Saad Al Omayrah (right)), https://rymarstudio.com/residentialarchitecture, https://www.behance.net/gallery/148907457/21-Villa-By-Depth-of-Field*

*Figure 2-14- Using louvers as architectural pattern in contemporary designs (Condomínio Terras de Toscana, by Lima Arquitetos (left), CB hoise, Indonesia, by Studio Avana (right) - http://www.limaarquitetos.com/projetos/residencia-br/, https://www.behance.net/gallery/98916083/CB-House*

*Figure 2-15- Using skewed and recessed elements to highlight openings (Villa in UAE by Nisreen Kayyali (left), Viewpoint House, Quezon City, Philippines by Jim Caumeron Design (right)) - https://www.instagram.com/nisreenkayyali/, https://www.archdaily.com/951932/viewpoint-house-jim-caumeron-design?ad_medium=gallery*

These motifs contribute to the visual language of architecture and are often used to communicate cultural, historical, or artistic references. Architects use motifs to create a sense of unity and coherence in the design, tying together various elements of a building into a harmonious whole.

**Summary**

In this chapter, the definition of architectural form is explored tracing its origin to the Latin term equivalent to idea. Over time, architects expanded this definition to encompass the intersection of mass and space, highlighting the multifaceted nature of forms beyond mere aesthetics. From ancient times to the present, theories on form have evolved, reflecting shifts from epistemological to methodological perspectives due to societal and economic developments. Geometry has played a crucial role, with historical roots linking it to divine principles, now utilized for rationalizing designs amidst technological advancements.

Conceptual elements such as points, lines, surfaces, and solids were analyzed, with points deemed particularly influential in form development due to their ability to establish relationships and parametrically define elements. Visual elements and design strategies like form, size, proportion, repetition, rhythm, harmony, and unity were discussed for their impact on form perception and design aesthetics, especially with advancements in digital fabrication techniques facilitating patterned facades and forms.

Form making, characterized by subjective and intuitive aspects, was juxtaposed with form finding, which relies on mathematical rules for problem-solving. Architectural patterns, driven by problem-solving approaches within specific contexts, were contrasted with architectural styles focused on compositional elements' formation. Notable motifs, including architectural orders of the Greeks, Gothic tracery, Moorish arches, Greek key patterns, Art Deco zigzags, and modernist grids, were analyzed for their significance and influence on architectural styles.

Contemporary architectural motifs and styles, observed in villas and residential buildings, exhibited recurring patterns such as exposed slabs, recessed glazing facades, L-shaped terraces, repeated solid blocks with openings, stripped louvers, and skewed recessed blocks. These motifs reflect cultural, religious, and aesthetic considerations, shaping architectural compositions.

In the subsequent part, these motifs will be incorporated into designed models to educate AI on architectural aesthetics and styles.

# Chapter 3: Coding in computational design: A base for utilizing AI in architectural form finding

**Preface**

Computational design in architecture utilizes computational tools and processes, including algorithms, scripting, and programming, to inform, generate, and optimize solutions. This approach often intersects with parametric design but encompasses a broader range of techniques. Parameters such as wall length, window size, and spatial relationships influence spatial experience, aesthetics, environmental interaction, and resource consumption. Architects' evolving expertise shapes their unique design approaches.

Parametric design translates architectural models into parameters and defines their relationships. Changing one parameter can alter others due to their interconnections. Architects use data types like numbers, booleans, or strings to create parametric models, which can include regulations like setbacks and heights. This algorithmic thinking allows for easy adjustments and maintains design consistency (Jabi, W., 2013). Software like Grasshopper for Rhinoceros3d and Dynamo for Revit arrange models as interconnected algorithms.

In September 2007, Rutten, D. developed the "Explicit History" plugin for Rhinoceros3d, which created a visible history of operations. This evolved into Grasshopper3d, enabling visual coding and forming the basis of parametric design with real-time modifications. Similar visual programming languages followed, including Dynamo for Revit, Marionette for Vectorworks, Param-O for Archicad, VizPro for Sketchup, and SIII for Blender, enhancing modeling capabilities (Sawantt, S., 2021).

Coding, defined as instructing a machine to perform tasks, underpins architectural software by transforming geometry through mathematical operations. Mastery of a software's coding language allows architects to use the software more effectively, reducing user interface biases. Coding provides tools like iterations, conditional statements, and extensive libraries for tasks such as data visualization and array manipulation, expanding design possibilities. Coding enables architects to perceive geometry as containers of information, enriching their design understanding.

This chapter discusses computational design as a design thinking approach, exploring its roots and presenting a taxonomy of methodologies. It reviews the generative form-finding approach, analyzes architectural forms as information and examines algorithm formation based on form elements, advocating for coding over visual programming languages. Additionally, the chapter provides an overview of coding practices, software functionality, and the importance of coding for its power and freedom, avoiding biases in visual programming and conventional modeling.

## 3.1 Computational design thinking

Computational design in architecture integrates advanced computational capabilities to automate, parallelize, and enhance various aspects of the design process. It enables architects to efficiently manage information, incorporate changes seamlessly, and explore diverse design possibilities through automation and algorithms, aligning with contemporary architectural demands.

Oxford Dictionary defines computation as "the action of mathematical calculation" and "the use of computers, especially as a subject of research or study." The Cambridge Advanced Learner's Dictionary & Thesaurus adds that it is "the act or process of calculating an answer or amount by using a machine." In architecture, Oxman (2006) describes computational design (CD) as design processes that fully utilize computers for their computational abilities rather than as electronic drawing boards. Terzidis (2006) defines CD as the entire process leading to a final result through digital tools. Thus, CD can be expressed as a design process leveraging computational capabilities through various activities (Caetano, I., et al., 2019) including automating design procedures through deduction, induction or abstract, parallelizing design tasks by breaking down the design process into smaller, more manageable components that can be processed simultaneously (figure 3-1), incorporating and propagating changes and assisting in form-finding processes.

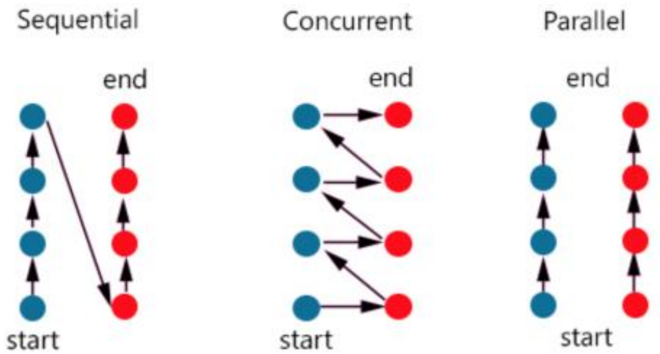



*Figure 3-1- Sequential vs. Concurrent vs. Parallel Design Processes (https://www.linkedin.com/pulse/concurrency-vs-parallelism-2-sides-same-coin-khaja-shaik-/)*

According to Menges, A., and Ahlquist, S. (2011), computational design (CD) is crucial because it shifts the perception, purpose, and production of form by utilizing information processing and interactions between elements, emphasizing systems thinking over by-element thinking. Systems thinking views every aspect of a form as part of a hierarchical structure of components. Menges and Ahlquist argue that computation is not necessarily related to computer use and should not be confused with computerization, which involves automation and digitization. They assert that computational methods can codify, analyze, systematize, and synthesize mental processes without digital tools.

Aristotle's definition of 'holism' underpins the understanding of systems as wholes greater than the sum of their parts. Descartes emphasized understanding processes through simple causalities, while Christopher Alexander highlighted that a system's overall behavior results from the interaction among its parts. Understanding these interactions is critical (figure 3-2).

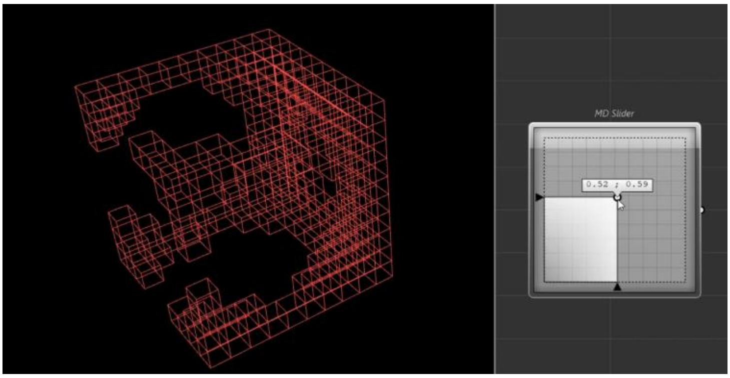



*Figure 3-2- Interaction between Components Forms the Whole Design Idea*

Glenn Wilcox argues that design computing views geometric forms as containers of information, not just shapes. Thus, a simple box is perceived not merely as a geometric object but as an entity with characteristics like height, length, and width. (figure 3-3 (left). In computational design thinking, a box is viewed as a container of information defining its characteristics and relationships with other architectural elements (figure 3-3 (right)). This box consists of points that determine distances between each other and other elements. Each surface of the box has attributes like color and texture, along with a center point. The box's center point and each point's coordinates (X, Y, Z) provide valuable data for spatial positioning. Extracting this information facilitates transformations and establishes relationships with other objects and their components.

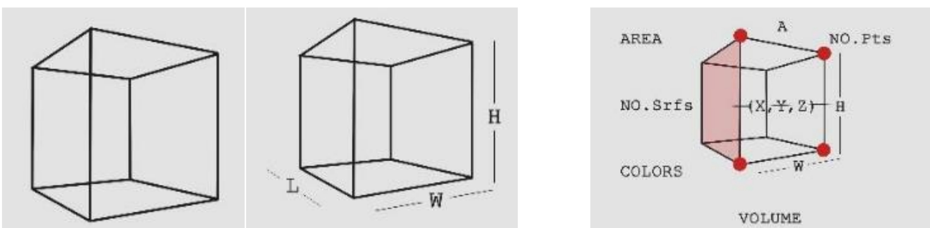



*Figure 3-3- (left) A box as geometry vs. a box with geometric existence characteristics, (right) A box is considered a container of many information rather than a simple geometry.*

Thinking about architectural elements through the lens of computational thinking, adds many layers of control and freedom to the way the elements are interacting to form spaces and architectural forms through the fast-developing tools in computers. Additionally, extracting those relations as parameters can serve as a basis for utilizing different ML in the form finding process.

### 3.2 The roots of computational design

The influence of geometry, proportion, and numerical principles in architecture dates back to ancient times (Burry, M., 2010). Vitruvius is credited with establishing the Vitruvian Triad—Firmitas (Durability), Utilitas (Utility), and Venustas (Beauty)—as the fundamental principles of architecture. Vitruvius emphasized the importance of geometric ideals, particularly Order and Arrangement (Ordinatio). He highlighted that the thoughtful organization of architectural elements creates a cohesive and visually pleasing composition, contributing to the overall harmony and effectiveness of the design.

The concept of the golden ratio, often denoted by the Greek letter phi (φ), has been known and utilized since ancient times, and it is not attributed to a single individual. The golden ratio is an irrational number, approximately equal to 1.618033988749895, and it appears in various mathematical and natural contexts. It is often expressed as the ratio of two quantities, where the whole is to the larger part as the larger part is to the smaller part.

The term "golden ratio" itself is relatively modern, coined in the 19th century. Mathematicians and artists throughout history, however, have been aware of and fascinated by this ratio. Ancient Greek mathematicians, including Euclid, explored the mathematical properties of the golden ratio, and it has been observed in the architecture of ancient civilizations, such as the Parthenon in Athens. The Italian mathematician Leonardo Fibonacci, in his "Liber Abaci" (1202), introduced the Fibonacci sequence, which is closely related to the golden ratio. However, it's important to note that while Fibonacci popularized the sequence, the golden ratio itself was known and used before his time.

Le Corbusier introduced the concept of "regulating lines" in his design principles. As observed in figure 3-4, these lines served as a framework for organizing and proportioning buildings. They were based on mathematical principles and were intended to provide a rational and harmonious basis for architectural composition. According to Le Corbusier, regulating lines, be they circular, square, or linear, manifest as helpful and regular points that intricately bring diverse elements together. The placement of angles in a precise manner is underscored as a mechanism for uniting varied qualities of architectural elements, fostering a sense of order and cohesion.

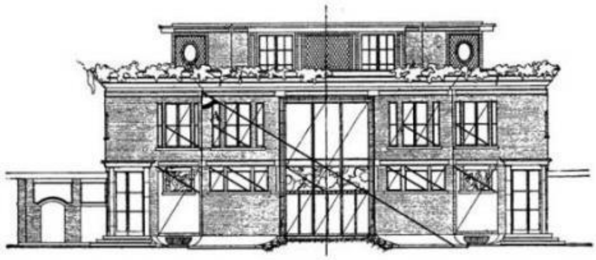


*Figure 3-4 – Villa Design by Le Corbusier (https://melissabilgecelik.wordpress.com/2018/10/31/regulating-lines-le-corbusier/)*

Additionally, Ludwig Mies van der Rohe frequently employed modular grids as a foundational design principle to manifest his commitment to simplicity and order. The modular grids (figure 3-5) provided a framework for organizing elements and spatial configurations, enabling rational and flexible compositions. Mies's grids balanced order with adaptability, accommodating various functions precisely. The Barcelona Pavilion and the Farnsworth House exemplify the enduring impact of his modular grid system on modern architectural design.

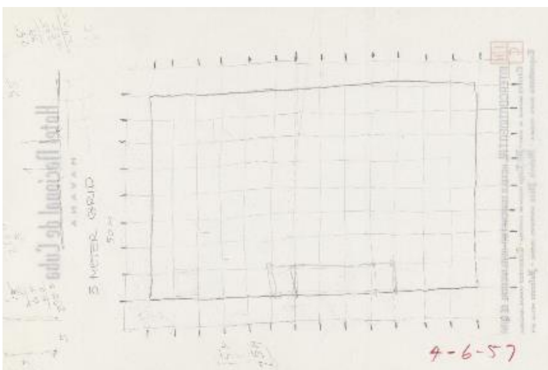



*Figure 3-5- Graphite and red pencil on verso of Hotel Nacional stationary by Mies Van der Rohe (https://www.moma.org/collection/works/87415)*

Today, the intensive use of CD is seen in the work of many architects who apply generative design techniques and optimization especially in the form finding process. Numbers and computation became a very important aspect in designs as will be discussed in section 3-4.

### 3.3 A taxonomy of computational design terms

The emergence of computation-based approaches in design has seen widespread adoption among architects and design professionals in recent decades. However, diverse terminologies such as computational, parametric, generative, algorithmic, performance-based, and evolutionary design have led to ambiguity and hindered communication within the field. This variability reflects the dynamic nature of computational design, rooted in a rich history dating back to the 1960s, influenced by pioneers like Ivan Sutherland (Caetano, A., et. Al, 2019).

Sutherland's concepts of design variation and parametric instances catalyzed the shift toward computational design. In the 1970s, efforts to formalize CD emerged, leading to its recognition as a distinct field within architecture in the 1980s. By the 1990s, computational design had solidified its presence, with dedicated conferences and journals. However, in the last two decades, it has evolved beyond automating drafting tasks to encompass diverse computation-based methods.

Contemporary computational design integrates techniques like building simulation, evolutionary optimization, and novel fabrication methods, driving innovative design approaches. As computational design continues to evolve, it remains at the forefront of architectural exploration, shaping the conceptualization and realization of built environments. In this section different CD related terms are reviewed, defining how each approach is involved in CD including CAD, parametric design, generative design, algorithmic design, and other related terms.

Computer-Aided Design (CAD) emerged in the late 1950s and early 1960s, with Ivan Sutherland's "Sketchpad" in the 1960s introducing interactive graphic design systems. The 1970s saw the development of commercial CAD systems like DAC-1 and CALMA for electronic design and manufacturing. The 1980s brought 3D CAD and the rise of desktop workstations, with Autodesk's AutoCAD becoming widely used in 1982. CAD became mainstream in the 1990s, with standardization efforts making it accessible across industries. It aids in creating, modifying, analyzing, and optimizing designs, improving workflow efficiency, quality, and documentation, and contributing to manufacturing design databases.

The history of digital design (DD) traces back to the 1950s and 1960s when early digital computers were primarily used for scientific and military purposes. Ivan Sutherland's "Sketchpad" in the early 1960s laid the foundation for digital design, marking the inception of digital tools in the design process. In the 1970s, computer graphics emerged, enabling designers to manipulate and visualize images on digital displays. Autodesk's AutoCAD in 1982 played a pivotal role in the widespread adoption of digital design tools. After that, the 1990s witnessed the democratization of digital design with the rise of personal computers and software like Adobe Photoshop, Illustrator, Rhino, and 3ds Max.

In the 21st century, the integration of digital design into various disciplines accelerated, exemplified by the rise of parametric design and algorithmic modeling. Today, digital design encompasses various fields from graphic and web design to product design, animation, and virtual reality. Advancements in technologies like augmented reality, AI, and generative design continue to shape the landscape of digital design, empowering designers to bring their visions to life in unimaginable ways.

Caetano, A, et. Al., 2019 argue that parametric design (PD) in architecture utilizes parameters and algorithms to create adaptable architectural forms, offering a wide range of design possibilities. Moretti (1971) describes PD as investigating relationships between design dimensions, while Kalay (1989) focuses on dynamic geometric representations. Szalapaj (2001) emphasizes geometric constraints, and Kolarevic (2003) views PD as declaring design parameters rather than specific shapes, allowing for multiple solutions. Eggert (2004) stresses PD's optimization capacity, while Schumacher (2008) sees it as a contemporary architectural style.

Woodbury (2010) highlights PD's associative nature, and Elghandour (2014) views it as code-based design. Zboinska (2015) categorizes PD under Algorithmic Design, emphasizing algorithmic processes. Zarei (2012) subdivides PD into conceptual modeling and construction/manufacturing categories.

Parameters enable designers to establish relationships and constraints governing the design, allowing exploration within a coherent system. Janssen and Stoufs (2015) categorize PD into object, associative, data-flow, and procedural modeling, providing comprehensive tools for designers across the design process.

Soleimani (2019) advocates for integrating parametrics into architecture programs through three interconnected approaches. Firstly, the system-based approach emphasizes studying architecture as complex subsystems rather than individual objects, engaging with spatial, material, social, and structural elements for harmonious coexistence. Secondly, the algorithmic, rule-based approach promotes computational thinking, using algorithms to create active relationships between design intent and outcome, yielding alternative design possibilities. Lastly, the interdisciplinary approach encourages architects to draw from diverse disciplines like philosophy, biology, mathematics, and computer science to address evolving challenges creatively. These approaches serve as essential pillars for the transformative integration of parametric design in architectural education.

Generative design (GD) employs computational systems to autonomously explore and generate potential solutions, akin to nature's evolutionary mechanisms. It surpasses the autonomy of Parametric Design (PD) by utilizing more autonomous algorithmic descriptions (Caetano, A, et. Al., 2019). GD systems, as defined by Mitchell (1977), generate solutions to design challenges without continuous direct input from the designer. Fischer and Herr (2001) characterize GD as a methodology where designers interface with generative systems, exploring and evolving solutions through computational means. Frazer (2002) compares GD to evolutionary processes in nature, highlighting its dynamic and iterative nature. Krause (2003) notes GD's autonomy in creating architectural structures or spaces, while McCormack (2004) describes it as generating complex designs from simple specifications. Bukhari (2011) positions GD as a subtype of Algorithmic Design (AD), utilizing algorithms to produce diverse solutions. These perspectives underscore GD's dynamic, algorithmically driven nature and its capacity to autonomously evolve and fulfill design criteria.

Algorithmic design (AD), according to Caetano, A, et. Al., 2019, leverages algorithms to generate models, establishing a clear correlation between the algorithm and the resulting design. This transparency enables users to trace and understand how different elements of the model are generated. AD provides generative capabilities through algorithms and enhances transparency and understanding throughout the design process. An algorithm, as defined by the Cambridge Dictionary, is a set of mathematical instructions or rules to solve a problem. Terzidis (2003, 2004) describes AD as generating space and form through rule-based logic inherent in architectural programs and language. Bukhari and Caldas (2011, 2008) note that AD includes both generative design and evolutionary design methods, employing fitness functions to guide the search process. Oxman (2017) emphasizes the procedural nature of AD, involving the explicit coding of instructions to generate digital forms. Zboinska (2015) views AD as a paradigm built upon Parametric Design (PD) tools, utilizing simple rules and relationships to produce complex geometries. Together, these perspectives define AD as a versatile paradigm encompassing various computational design methods.

Caetano, A, et. Al., 2019, suggested a conceptual overlap and inconsistent use of terms related to CD, with a specific focus on PD, GD, and AD. They illustrated this conceptual overlap using a Venn diagram (figure 3-6), indicating that AD is a subset of GD and shares a non-empty intersection with PD.

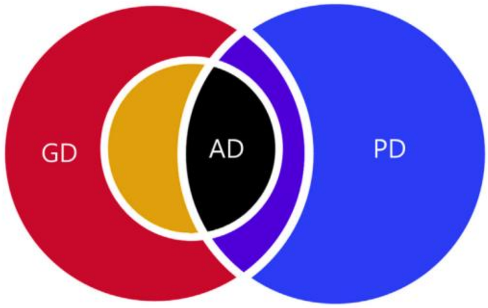



*Figure 3-6 - Conceptual representation of the terms' extension regarding the CD paradigm, by Caetano, A., et. Al., 2019, Computational design in architecture: Defining parametric, generative, and algorithmic design.*

Caetano, A, et. Al., 2019, argue that there are multiple cases of overlaps:

1- AD (Both GD and PD): Designing an algorithm that generates a facade based on a set of parameters like the dimensions, size, and distribution of different elements.

2- GD, PD, but not AD: Designing through optimization, however, the relations between the parameters and the optimization mechanism is difficult.

3- GD but neither PD nor AD: Using cellular automata in design where the rules are not parametric, and the outcome is nearly impossible to directly infer from the rules of the automaton.

4- PD but neither AD nor GD: Designing an element such as a wall allowing users to change the parameters without requiring explicit use of algorithms.

5- GD, AD, but not PD occurs in digital fabrication when a computer numerical control machine operates, executes a program that is often automatically generated and entirely non-parametric.

Last but not least, and according to Caetano, A., et. Al., 2019, "performance-based design" is the third most used term after PD and GD, but less prevalent than Parametric Design (PD) and Generative Design (GD). There is limited overlap with the term "performative design," which is not as commonly used. They presented a bar diagram illustrating the frequency of appearance of each CD-related term in the literature from 1978 to 2018.

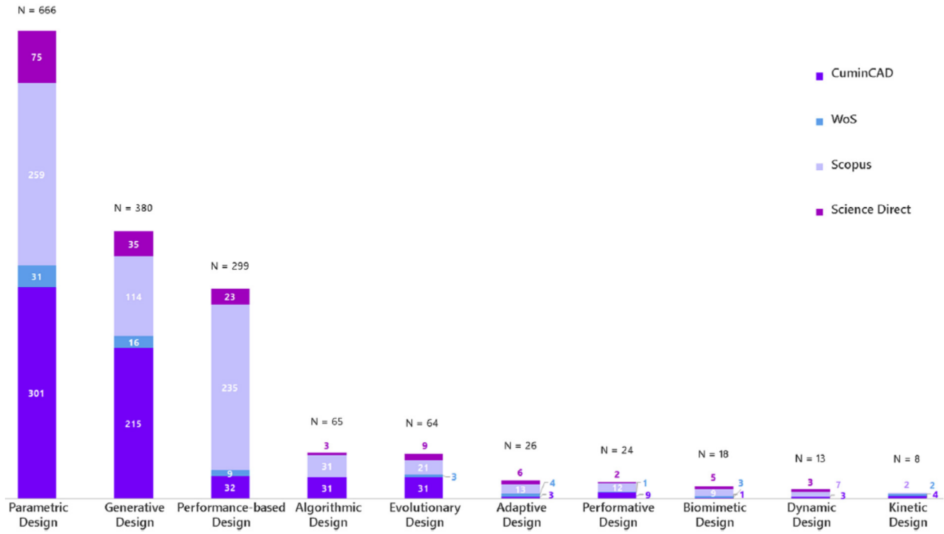



*Figure 3-7- Number of times each CD term appeared in the literature between 1978 and 2018 by Caetano, et. Al, 2019*

Overall, performance-based design integrates performance criteria throughout the design process, prioritizing specific outcomes over traditional standards. It's prevalent in architecture and engineering, focusing on achieving desired performance goals. Evolutionary design, adaptive design, performative design, and kinetic design are variations within this approach.

GD utilizes algorithms to generate designs autonomously, emphasizing computational processes to produce diverse solutions. AD, a subset of GD, emphasizes traceability between algorithms and outcomes. PD relies on parameters to describe designs, allowing flexibility and adaptability within a defined framework. Understanding these terms is essential for employing specific techniques and approaches in the design process.

## 3.4 Generative form finding

Generative form-finding is a design process rooted in rules or algorithms, often facilitated by software like Rhinoceros, Grasshopper, Dynamo, and scripting platforms. Eisenman's influence in the 80s and early 90s, inspired by Derrida's deconstruction theory, introduced techniques like overlay, fractals, and scaling, paving the way for contemporary generative design attempts. Greg Lynn further advanced the field with techniques such as NURBS and splines, leading to the emergence of 'blob architecture.'

With the advancement of computational and scripting tools, digital fabrication became more accessible, enabling explorations in paneling, optimization, simulations, and algorithmic design. Generative design leverages computers to explore solutions, sometimes through algorithms, and employs evolutionary optimizers to reach design goals. This approach shifts the focus from the final form to the underlying logic of design, externalizing the designer's intelligence into generative systems. Various tools like shape grammars, parametric variations, and evolutionary algorithms enable designers to encode rules and algorithms, guiding form generation. Additionally, newer tools explore randomness and chaos in form generation, broadening the spectrum of generative design possibilities. (El Iraqi, A., and El Daly, H., 2017).

According to El Iraqi, A., and El Daly, H., 2017, generative design systems can be broadly classified into two categories: linguistic generative systems where the emphasis is on encoding design rules and logic (syntax) in a language-like structure that govern and shape the design (semantics) and biological generative systems which draw inspiration from natural processes, particularly those related to evolution. Genetic Algorithms and Cellular Automata are examples of biological generative tools.

Linguistic generative systems includes shape grammars, developed by Stiny and Gips in 1972, which formalize rules for generating shapes or forms, particularly useful in architecture and urban planning, L-systems, which was introduced by Lindenmayer, model growth processes, especially for self-replicating structures like plants, and fractals, being complex shapes with self-similarity, generated through recursive algorithms and have applications in diverse fields for creating intricate and visually appealing forms.

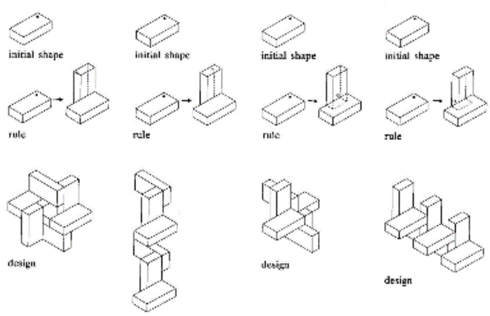



*Figure 3-8- Shape Grammars (MIT - Computational Design I: Theory And Applications - Fall 2005 Lecture 7)*

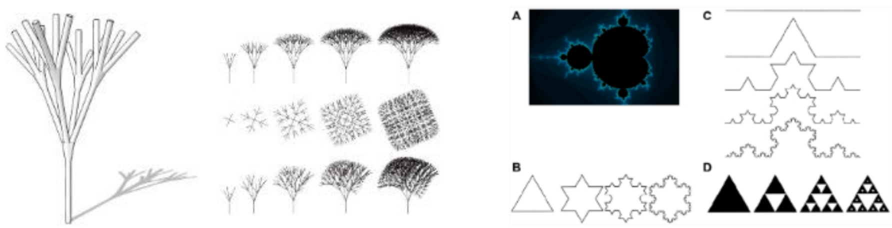



*Figure 3-9- Left: L-Systems (Chiu, P., 02015, The Structure of L-System), Right: Examples of geometrically self-similar fractals. (A) The Mandelbrot set. The " curve " (B) and the " snowflake " (C), described by Niels Fabian Helge von Koch (1870–1924), and the " Sierpinski triangle " (D), described by the mathematician Waclaw Sierp – Di Leva, A., et. Al, 2013, Fractals in the Neurosciences, Part I: General Principles and Basic Neurosciences*

Biological generative systems include genetic algorithms which mimic natural selection to find optimal solutions by creating populations, evaluating fitness, and using genetic operators like crossover and mutation. Solutions are represented as individuals (phenotypes) with encoded parameters (genotypes), genes, alleles, and chromosomes. GAs evolve over generations toward optimal solutions. They also include Cellular automata (CA) which are discrete computational models where cell states evolve based on rules determined by neighboring cells (Robert J., K., 2002). CA is used in generative design to create complex patterns and simulate dynamic systems.

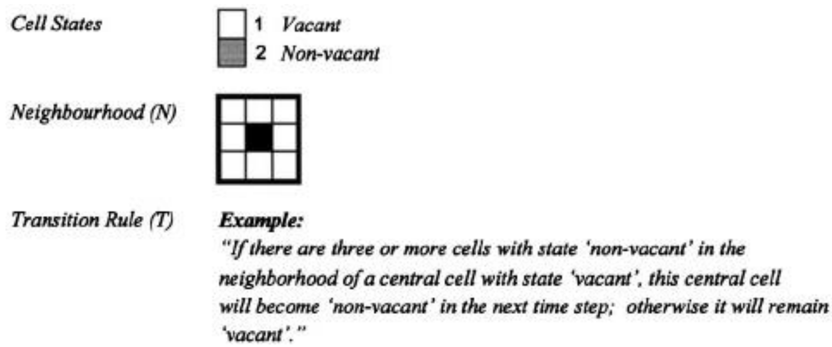



*Figure 3-10- Components of Generic Cellular Automaton - Leao, S., et. Al., 2004, Simulating Urban Growth in a Developing Nation's Region Using a Cellular Automata-Based Model*

In essence, the generative form finding represents a pivotal moment in design evolution, where technology becomes a co-creator, pushing the boundaries of what is conceivable and achievable in architecture and engineering. Algorithms and parametric models drive the creation of complex, self-adaptive structures. This union facilitates not only the exploration of diverse design possibilities but also the optimization of forms based on performance criteria.

### 3.5 Architectural forms as information

The use of computer-aided tools in architectural design, as highlighted by Jabi (2013), has facilitated the creation of more complex designs, especially with the emergence of scripting tools that offer architects greater control over design elements. Architectural forms serve as repositories of valuable information in computational design, encompassing geometric, material, environmental, cultural, and experiential data. This information influences design decisions and can be utilized to optimize designs for sustainability, functionality, and user experience. Each parameter of a building influences its spatial aspects and user interaction, underscoring the importance of understanding these factors. Furthermore, translating architectural information into datasets for ML algorithms enables pattern recognition and prediction, driving innovation and efficiency in design processes. Overall, treating architectural forms as information enhances design thinking and enables a deeper understanding of the relationship between physical space and digital design. And the best way to deal with large quantities of information is through algorithms.

The advent of parametric design software such as Grasshopper for Rhino and Dynamo for Revit shifted the way architectural modeling tasks are done from a conventional method of drawing to creation-by-algorithms. Algorithms require input data/parameters and steps/operations/conditions on those parameters to reach a final output. For instance, a simple example of an algorithm for solid wall creation requires inputs such as points, a rectangle, or a surface. In the case of a rectangle, an operation of converting it to a surface, and then extruding it and in case of a surface, only extrusion process is required. Now, if an architect needs to adjust the wall parametrically, they can change the rectangle dimensions or the extrusion height. Transformations are operations applied on the shape that could change its position, angle, scale, etc. and each is considered an operation/rule applied within the algorithm to reach the final output. Copying the wall, connecting walls and slabs, opening walls, transforming items to respect setbacks, and more are examples of rules applied to the same algorithm to reach a building rather than a wall. Afterwards, adjusting any parameter within the same framework of the algorithm maintaining its structure will be easy. However, the number of parameters can be critical because regardless of how many parameters an architect adds to the algorithm, relations between those parameters should be kept clear in order to maintain the algorithm's readability and function. And so, the way an architect can algorithmically decompose a building and find the proper relationships between its components can be very complex. Especially, with a large number of components each related to another. But what is guaranteed is that each design will be a product of a well-structured and connected algorithm that maps exactly how the architect thinks and what decisions have been made. And there comes the importance of parametric design approach where an architect could use hundreds of parameters that together shape the form while being interrelated directly so that changing a parameter could affect other parameters.

### 3.6 Coding as a Practice

"Coding" typically refers to the act of writing code, which is a set of instructions written in a programming language. This can involve translating a specific algorithm or set of tasks into a language that a computer can understand and execute. Coding is often considered a more casual term, and it can be used to describe both the broader process of writing code and the specific act of writing individual lines or blocks of code. In the Oxford Languages Dictionary, the term ‘code’ is defined as a ‘system of words, letters, figures, or symbols used to represent others, especially for the purposes of secrecy’ and ‘program instructions.’ Also, the term ‘coding’ is defined as ‘the process of assigning a code to something for classification or identification’ and ‘the process or activity of writing computer programs’ (Oxford Languages Dictionary).

From these definitions, coding is not a process that is exclusively related to computers. Human mental processes could be done through coding. In fact, coding becomes a general activity that involves ciphering data, analyzing, and synthesizing it to solve problems. In qualitative research, coding is essential for organizing and analyzing data, assigning labels to segments of qualitative data to identify patterns and insights.

While coding and programming are often used interchangeably, programming encompasses the entire software development process, including coding, problem-solving, testing, and maintenance. It requires a comprehensive understanding of the software development life cycle. In practice, coding involves writing instructions for software to perform specific tasks, often hidden behind a graphical user interface (GUI). When a user interacts with a GUI, such as clicking a button, it triggers a series of processes between the front end (GUI) and back end (software logic), ultimately executing the desired action. This process could be broken down simply into the following steps: user interaction in the GUI triggers event handling in the front end, where the software captures and processes the user's action. This information is communicated to the back end, which determines the appropriate response based on the user input. The software's logic executes the action, translating high-level code into machine code for the computer's CPU to process. If necessary, the software may undergo compilation before execution. The updated GUI provides feedback to the user, indicating that the action has been completed. This seamless process ensures intuitive and responsive user experiences in software applications.

The GUI triggers actions in the software's logic, leading to the generation and execution of machine code by the computer's hardware. The process varies by programming language: C/C++ directly translates code to native code, Java to bytecode, C# to an intermediate language, which is then Just-In-Time (JIT) compiled to native code, and Python to bytecode executed by its interpreter. This variation affects how code is executed and optimized.

Machine code, written in binary or hexadecimal notation, is specific to a computer's architecture and operating system. Developers typically use higher-level programming languages and rely on compilers or interpreters to generate machine code. The actual instructions executed depend on factors like the programming language, operating system's API, and hardware architecture. Lower-level languages like C/C++ may use system-specific functions to interact with the OS, while higher-level languages like Java/Python delegate interaction to the runtime environment or interpreter. Understanding coding principles empowers users to leverage machines fully, fostering creativity in task execution and potentially innovating new functions.

### 3.7 How Modeling Software Work

Programs, or software, are sets of instructions enabling computers to perform tasks, essential for computing's functionality. Programmers and software developers design, create, and maintain software, shaping applications that empower computers. Modeling software like Rhinoceros, 3ds Max, Revit, and Maya interact with computer hardware to create and manipulate 3D models. Developed using high-level languages such as C++ or C#, they employ frameworks for GUI, event handling, and rendering. GUI facilitates user interaction, with inputs processed through event handling. Core functionalities, like surface modeling, employ complex algorithms implemented using high-level languages. 3D rendering engines interface with the GPU for realistic visuals. File operations use libraries for compatibility with standard formats, integrating with OS APIs for tasks like file management. Custom scripting and plugins, often in Python, extend functionality. Memory management and multi-threading considerations optimize performance, making them indispensable tools in design, animation, architecture, and engineering.

***Modeling and rendering geometry***

Geometry modeling algorithms in 3D software, like Bezier surfaces and NURBS, use mathematical representations to define and manipulate geometry. Implemented in languages such as C++, C#, or Python, these algorithms compute points on the geometry based on mathematical formulations, often employing techniques like De Casteljau's algorithm for Bezier surfaces. Rendering processes, facilitated by OpenGL or DirectX, utilize GPU acceleration for real-time visualization, incorporating shading and lighting algorithms for realism. The GUI allows user interaction for operations like selection, translation, and scaling, with event handling mechanisms triggering updates. Optimization techniques enhance performance with complex geometry. The rendering pipeline, used by libraries like OpenGL, converts 3D data into visual images, with shaders enabling custom visual effects for enhanced realism. Translation to machine code involves compiling high-level language code into CPU-executable instructions, typically written in languages like C++.

The rendering pipeline encompasses a series of mathematical operations and algorithms translated into machine code instructions for efficient execution on a computer's hardware. This includes transforming vertices, applying view transformations, projecting coordinates, and clipping to ensure visibility, followed by rasterization to determine pixel coverage. Vertex and fragment shaders, compiled from high-level shading languages like GLSL or HLSL, handle shading operations in parallel on the GPU. Texture mapping involves calculating texture coordinates and sampling, while depth testing compares pixel depths for drawing order. Alpha blending logic is applied for transparency, and frame buffer operations manage pixel storage and display updates. Optimization techniques are employed during compilation to generate efficient machine code, leveraging parallelism, particularly in GPU programming, for simultaneous execution of shader operations on multiple vertices or fragments.

The compilation process translates high-level code into an intermediate representation like bytecode or assembly code. During linking and loading, this representation is further translated into machine code specific to the CPU or GPU architecture, resulting in an executable program runnable on the hardware.

***Software documentation***

Software documentation is a comprehensive set of written materials that serves to describe, explain, and guide various aspects of a software system. The information in software documentation is often divided into task categories, including evaluating, planning, setting up or installing, customizing, administering, using, and maintaining. Different types of documentation play crucial roles throughout software development life cycle and there are mainly two types which are internal and external software documentation. Internal software documentation serves as a valuable resource within a company including administrative documentation which includes administrative guidelines, roadmaps, and product requirements and developers' documentation which offers clear instructions to developers on how to build the software. On the other hand, external documentation includes user documentation which provides guidance on product usage to the end-users, developer documentation which focuses on system-related details including how to invoke the API, and just-in-time documentation used where immediate support is needed for customer-facing queries, minimizing the need for users to refer to additional documents or FAQs.

***APIs and SDKs***

API (Application Programming Interface) and SDK (Software Development Kit) are essential tools in software development, each serving distinct purposes. An API defines rules for software component interaction, enabling developers to access functionality without exposing internal details. On the other hand, an SDK is a comprehensive package containing tools, libraries, and resources for building applications on specific platforms or frameworks. While APIs specify interaction rules, SDKs provide a complete development environment, including APIs, documentation, sample code, tutorials, and other tools.

## 3.8 Visual Programming Language

Visual programming languages (VPLs) use graphical elements like icons and symbols to represent programming logic, aiming to simplify coding and make it accessible to non-programmers. The concept of VPLs has evolved over decades, originating from the early development of graphical user interfaces. In architecture, visual programming emerged alongside computational design and parametricism, allowing architects to explore algorithms and dynamic parameters in design processes. Grasshopper, integrated with Rhino 3D modeling software, played a significant role in popularizing visual programming by offering a user-friendly interface for creating parametric designs. This approach facilitated interdisciplinary collaboration and streamlined architectural practices. Figure 3-11 shows an algorithm that creates a box using domains in the X, Y, and Z directions, and then moves it to the Z direction.

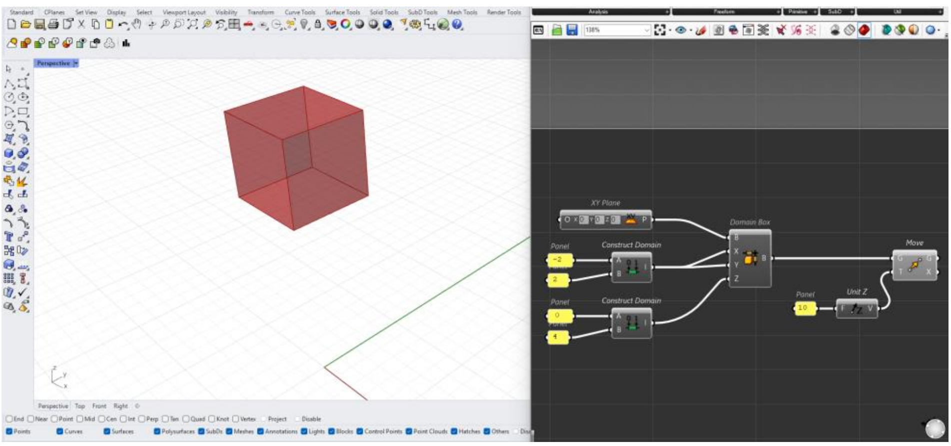



*Figure 3-11- Domain Box Creation as An Algorithm*

In the background of such an algorithm, each component runs a certain line of code that executes the function. The XY Plane component runs a 'Plane' struct property 'Worldly', the domain box component runs a 'Box' Class constructor, the construct domain component runs an 'Interval' Struct, the unit Z component runs a 'Vector3d' struct 'ZAxis' property, and the move component runs a 'Translation' method from the 'Transform' struct in RhinoCommon. The required parameters such as domain values and unit Z, vector value, etc., are considered input variables in the code.

Conventional modeling techniques in architecture often involve manually drawing shapes and structures, which can be non-algorithmic and non-parametric. This process may lead to inefficiencies when editing geometry and may require recreating geometry from scratch for complex tasks. Visual Programming Languages (VPLs) offer a graphical representation of programming concepts, allowing users to intuitively understand and manipulate program logic.

While VPLs enhance accessibility to programming, they may face challenges in expressing complex algorithms and handling large-scale tasks compared to text-based languages. They could lead to misrepresentation of a project algorithm due to the huge number of components on the screen that are connected to each other with wires resulting in a very tedious and unarranged virtual working space as shown in figure 3-12.

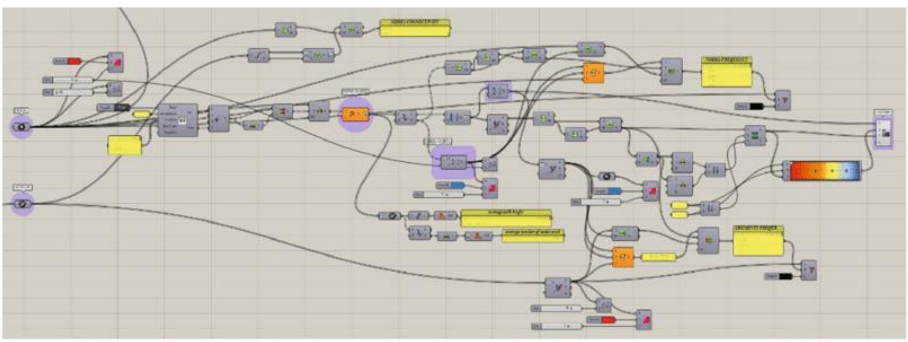


*Figure 3-12 - Grasshopper3d Definition Arrangement*

In summary, VPLs provide a graphical approach to coding, offering intuitive interfaces and visual representations of code structures. While they enhance accessibility, challenges exist, such as managing complexity in larger projects.

### 3.9 Bias in Modeling Processes and Leveraging Power, Freedom, and Spruceness of Coding

Bias in modeling processes arises from both software design and educational factors. Conventional modeling approaches limit creativity by dictating how geometry is built, often restricting users to predefined methods within the software's GUI. This bias impedes the ability to treat geometry as dynamic information, hindering the full utilization of mathematical operations in design. For example, in such software, architects may lack flexibility in constructing shapes or evaluating surfaces, leading to limitations in modifying and building upon geometry. Curve manipulation is particularly challenging, with divisions often resulting in kinks and disruptions to smoothness. Overall, conventional approaches constrain architects to predefined methods and limit their ability to fully leverage mathematical operations in the design process.

Parametric modeling addresses limitations in conventional approaches by enabling mathematical manipulation of geometry. However, educational bias persists as architects often learn software through predefined methods, hindering creative thinking. Users may favor easier modeling approaches, leading to a narrow perspective on software capabilities. Additionally, tutorials often focus on tools rather than mathematical principles, further limiting understanding. Overall, bias in modeling software restricts both thinking and modeling processes for architects.

Architectural design through coding empowers architects to innovate by fostering computational thinking, facilitating the creation of complex forms, optimized spatial layouts, and innovative design solutions.

***The power of coding*** lies in automating repetitive tasks, speeding up design iteration, and exploring various possibilities. Functions like iterations and conditionals are fundamental, offering control over code flow and flexibility. Iterations, via loops, enhance efficiency by repeating tasks and processing data sets, crucial for automation and batch processing. Conditionals enable decision-making, executing code blocks based on conditions, vital for error handling and user interaction. Together, they form the foundation for dynamic and efficient coding, handling diverse scenarios and data precisely. Additionally, coding facilitates interactions with the operating system, simplifying tasks like exporting model data as photos or spreadsheets for use in other disciplines, reducing reliance on multiple software tools.

***The freedom offered by coding*** in architectural design liberates architects from traditional constraints, enabling them to express design intent algorithmically through parametric models. This dynamic approach fosters flexibility, facilitating efficient design modifications in response to evolving project needs or client feedback. Working with the modeling software's API empowers architects to freely explore its functions and modeling methods, aligning with project requirements and enhancing the design process.

***Rhinoceros3d SDK***

RhinoCommon, McNeel & Associates' cross-platform .NET plugin SDK for Rhinoceros3d, offers extensive capabilities for extending and integrating functionalities within Rhino. Primarily designed for .NET languages like C#, it provides a versatile API with a robust geometry library at its core. Beyond basic scripting, RhinoCommon enables task automation and the development of custom plugins, enhancing Rhino's native features with bespoke tools. Its cross-platform compatibility ensures seamless operation across various operating systems, and supported by an active community, developers can tailor solutions to diverse design and engineering needs effectively. Figure 3-13 shows the RhinoCommon API website exhibiting all of the namespaces and their different classes and structs.

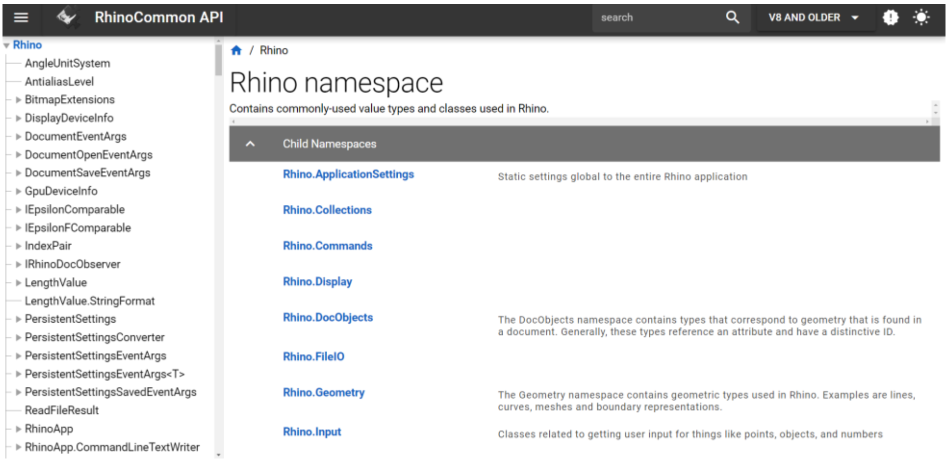



*Figure 3-13 – RhinoCommon API - https://developer.rhino3d.com/api/rhinocommon/*

To construct a box through coding using the API, the 'Box' struct offers six different constructors tailored for various scenarios. These constructors include methods for creating a box from a bounding box, copying another box, constructing it with a base plane and bounding box, using a base plane and a generic piece of geometry, requiring a base plane and a list of points (at least 2), and finally, one that needs a base plane and three intervals in each Cartesian coordinate (figure 3-14).

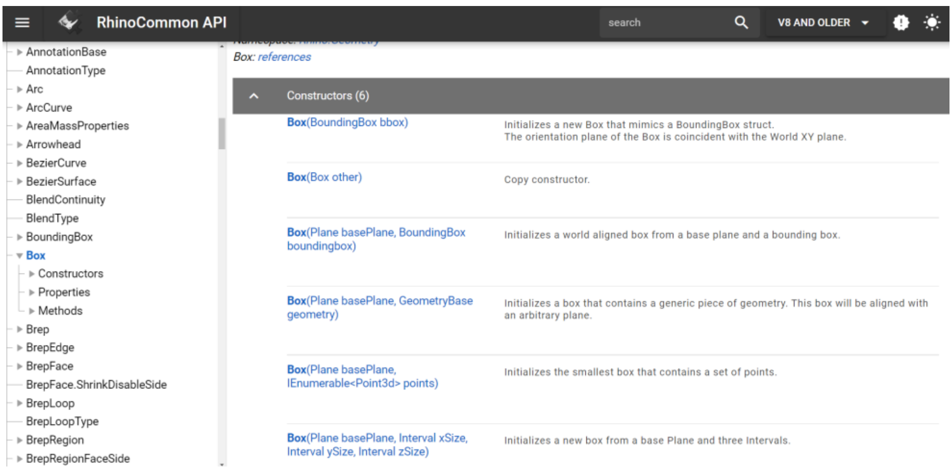



*Figure 3-14- Box Struct in RhinoCommon API - https://developer.rhino3d.com/api/rhinocommon/rhino.geometry.box*

Also, under the 'Box' struct, a list of properties (usually mathematical) related to the box are exhibited in order to gain all the possible information from it like its area, volume, orientation plane, center, X, Y, and Z intervals. Moreover, a set of methods are present for box geometries which facilitate getting more information about boxes like their corners, if it contains a certain point or another box, the closest point to a box, converting the box to a Boundary Representation (Brep) object, and transforming the box.

***The spruceness of coding*** in architecture refers to the cleanliness and efficiency achieved through well-organized and concise code. It emphasizes modular scripting, enhancing collaboration and sharing of design methodologies within the architectural community. This approach improves code readability, scalability of design solutions, and logical thinking skills for architects. In software like Grasshopper and Dynamo, a single coding component can create, modify, and transform every building component, resulting in a tidy virtual workspace. Figure 3-15 shows how tidy a virtual working environment can get with coding.

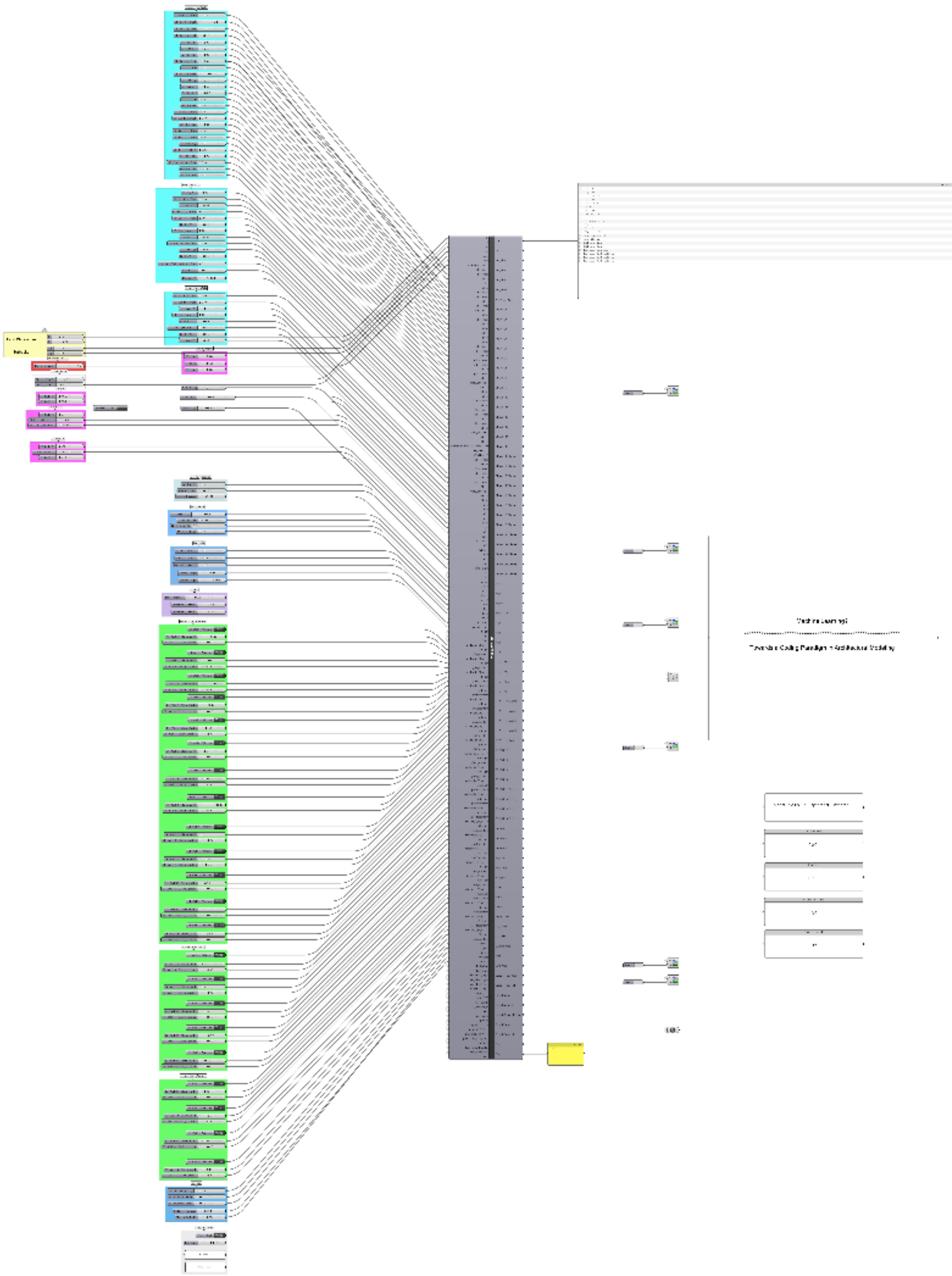


*Figure 3-15 - Parametric Villa C# Script Created using C# Component in Grasshopper- By the Author*

In general, integrating coding into architectural design modeling empowers architects with the computational tools needed to transcend traditional design boundaries. It provides the power to create intricate and responsive designs, the freedom to explore diverse possibilities, and the spruceness of well-crafted, efficient code. This symbiotic relationship between coding and architectural design opens new horizons for creativity, efficiency, and collaboration within the field of architecture.

## Summary

In conclusion, this chapter provides a comprehensive exploration of computational design as a transformative design thinking approach. It delves into the historical roots of computational design and presents a nuanced taxonomy that encapsulates the diverse approaches that have emerged under its expansive umbrella. The focal point of the chapter revolves around the in-depth examination of the generative form-finding approach, shedding light on its principles and applications in architecture. The collaborative engagement between designers and computational tools enables the externalization of design intelligence, fostering a dynamic and iterative design process. As designers increasingly leverage generative algorithms, shape grammars, and other computational techniques, the creative landscape expands, providing a rich platform for exploration, experimentation, and the realization of novel design solutions.

By analyzing architectural forms as a source of information for computational design, architectural forms become not only design outcomes but also repositories of data that inform and shape the computational design process. A crucial aspect discussed in the chapter is the translation of this information into algorithms. The formation of algorithms, rooted in the extracted information from architectural forms, serves as a pivotal step in the computational design approach. In the next part integrating coding, ML, and AI in the architectural design process is discussed.

This chapter delves into the evolution of parametric design and visual programming in architecture, tracing its roots back to the 'Explicit History' plugin for Rhinoceros3d software released in September 2007. The concept of 'visual coding' emerged, forming the basis for parametric design, where architects gain full control over parameters and real-time modifications. Coding, defined as the direct means to instruct machines, is explored as an integral part of architectural design and modeling. Architectural software communicates with machines through coding and mathematics, using transformations like altering coordinate systems to visualize geometry. Mastery of coding languages empowers architects to understand software mechanics deeply, reducing biases inherent in user interfaces. The chapter emphasizes the power of coding, incorporating built-in functions such as iterations and conditional statements for precise design control. Additionally, the integration of libraries in programming languages enhances the capabilities of architectural modeling, from data visualization to working with matrices and arrays.

In essence, coding is presented as a transformative practice in architecture, enabling architects to read geometry as containers of information rather than mere shapes. The chapter concludes by highlighting the importance of coding as a modeling approach, emphasizing its power and freedom to overcome biases in visual programming languages and conventional modeling methods.

# Part 2: Integrating AI in the Architectural Design Process (A Framework for Utilizing AI in Form Generation)

# Chapter 4 Artificial Intelligence and Machine Learning in Architecture

**Preface**

Today, AI and ML stand as pillars of transformative forces, reshaping the understanding of what machines can achieve. This chapter begins with the fundamental definition of AI. The definition of AI is multifaceted, encapsulating the development of machines and systems endowed with the capacity to perform tasks that traditionally require human intelligence. From rule-based systems to advanced neural networks, the breadth of AI's definition encompasses a spectrum of capabilities that continues to expand with technological advancements. Machine Learning, a subset of AI, forms the backbone of intelligent systems. It is the engine that enables machines to learn from data and improve their performance over time. The interplay between AI and ML is symbiotic, with ML providing the adaptive capabilities that empower AI to navigate dynamic environments.

Khean et al. (2018) highlighted architecture as one of the slowest industries to integrate AI and ML due to factors like traditional practices prioritizing craftsmanship and artistic expression, limited data availability, and the complexity of design. They underscored the importance of architect-AI interaction for favorable outcomes. This aligns with the "human-centered AI" approach, emphasizing collaboration between AI systems and human experts to enhance architects' capabilities and creativity, rather than replacing them.

In this chapter, the history of AI from the early philosophical musings to pivotal moments that have shaped the field is reviewed. In addition, generative AI (Gen-AI) and non-generative AI (Non-Gen-AI) are explored. The fundamental differences between systems designed for specific domains and those aiming to replicate human-like cognitive abilities across a spectrum of tasks is discussed. These distinctions have profound implications, not only in technical realms but also in ethical and societal dimensions. Moreover, some of the Gen-AI and non-Gen AI applications in architecture are reviewed and how these applications can be involved in the design process is discussed. Additionally, building on what was discussed earlier in chapter 1 regarding design thinking and architectural design process, the authenticity of Gen-AI products is questioned with a focus on generated images. Gen-AI is analyzed as a concept regarding how it could affect the design process, proposing a theory on how Gen-AI could fit in the process rather than dramatically changing it in a way contradicting with the essence of architectural design. After that, architectural visualization field and whether it directly affects the design process or not is discussed. Finally, how non-gen- AI could be integrated into the design process is explained.

## 4.1 AI Definition and History

AI refers to the simulation of human intelligence in machines that are programmed to think and learn like humans. It involves the development of computer systems capable of performing tasks that typically require human intelligence. These tasks include learning from experience (ML), understanding natural language, recognizing patterns, solving problems, and adapting to new situations.

AI can be approached through four perspectives, as outlined by Russell and Norvig (2010): thinking humanly, acting humanly, thinking rationally, and acting rationally. Thinking humanly, proposed by Haugeland (1985) and Bellman (1987), envisions AI as machines with minds, capable of human-like thought processes. Acting humanly, as described by Kurzweil (1990) and Rich and Knight (1991), focuses on creating machines that perform tasks requiring human intelligence. Thinking rationally, articulated by Charniak and McDermott (1985) and Winston (1992), explores AI as the study of mental faculties through computational models. Acting rationally, defined by Poole et al. (1998) and Nilsson (1998), views AI as the design of intelligent agents capable of exhibiting intelligent behavior. These perspectives converge on AI's core components: learning, reasoning, problem-solving, and creativity, drawing from diverse disciplines such as psychology, mathematics, linguistics, neuroscience, philosophy, and computer engineering.

Alan Turing's seminal 1950 article, "Computing Machinery and Intelligence," laid the foundation for AI, introducing the Turing Test and pioneering concepts like machine learning and reinforcement learning. Preceding this, Turing began discussing AI in 1947 lectures at the London Mathematical Society. In 1956, the Dartmouth workshop convened prominent figures like John McCarthy, Allen Newell, and Herbert Simon, marking the formal inception of AI as a field. McCarthy's creation of Lisp in 1958 revolutionized AI programming. The period between 1952 and 1969 witnessed key AI applications like the General Problem Solver (GPS) by Newell and Simon, and Arthur Samuel's checker AI player. The late 1960s and early 1970s saw a shift toward knowledge-based systems, exemplified by projects like DENDRAL and MYCIN, while the late 1970s marked the commercialization of AI. The mid-1980s experienced both progress, with neural networks challenging symbolic approaches, and setbacks, known as the "AI Winter." Subsequent years emphasized empirical experiments and real-world applications, with the late 1990s and early 2000s witnessing a shift to data-centric approaches. In the 21st century, AI advancements have focused on intelligent agents, integration with the Internet, and the pursuit of human-level AI, with ethical considerations driving research and development (Solomonoff, G., 2023).

In the evolution of computer science over the past six decades, there has been a notable shift from algorithm-centric approaches to a focus on the significance of data in AI development. This shift, highlighted by Yarowsky's 1995 work on word-sense disambiguation and Banko and Brill's 2001 study, emphasizes that the quality and quantity of available data may outweigh the importance of algorithm choice. Yarowsky's approach demonstrated achieving high accuracy without labeled examples by leveraging vast unannotated text corpora, while Banko and Brill's study showed that increased data volume can surpass algorithmic variations in performance. Further evidence from Hays and Efros (2007) illustrates how algorithmic performance improves with a larger collection of images, reinforcing the impact of data scale on AI outcomes. This data-driven paradigm suggests a potential solution to the "knowledge bottleneck" in AI, where comprehensive system knowledge is acquired through learning rather than manual knowledge engineering, given sufficient data for training algorithms. These developments signal a resurgence in AI applications, potentially marking the end of the "AI Winter" era and ushering in a new era of innovation across diverse industries, as acknowledged by Kurzweil's recognition of AI's pervasive integration into various sectors.

## 4.2 Types and Applications of AI

This section presents various types of AI based on its capabilities and functionality as well as different applications of AI.

***AI Types***

Biswal, A., 2023 categorizes AI into several types based on its capabilities, including Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI), Artificial Superintelligence (ASI), and Singularity. ANI, also known as weak AI, is specialized in performing specific tasks and encompasses the first and second waves of AI, involving expert systems, artificial neural networks, and data mining, among others. ANI systems excel in tasks like image recognition, speech recognition, natural language processing, recommendation engines, machine translation, and self-driving cars, but are limited to these tasks and lack generalization abilities. AGI, on the other hand, aims to mimic human-level intelligence across a broad range of tasks, possessing adaptability, learning capabilities, complex reasoning, and potentially self-awareness. ASI is a speculative type of AI that surpasses human intelligence in all aspects, potentially capable of recursive self-improvement. The Singularity refers to AI achieving autonomy and intelligence to break free from human control, leading to an intelligence explosion. The development of AGI is seen as a prerequisite for the Singularity, although there is uncertainty surrounding its occurrence and timeline.

AI types based on functionality include various categories, each delineating specific attributes and capabilities. Reactive Machines, exemplified by IBM's Deep Blue, operate solely on current input data without drawing from past experiences, limiting their adaptability beyond predefined tasks. Limited Memory AI strikes a balance by incorporating short-term memory, crucial for tasks like autonomous driving systems, exemplified by Mitsubishi Electric's advancements in this field. Theory of Mind AI endeavors to endow machines with the ability to understand and attribute mental states to themselves and others, showcased by creations like Kismet and Sophia. Lastly, Self-Awareness in AI pertains to recognizing one's own mental states, from basic understanding to nuanced awareness, with implications for enhancing human-machine interactions and system performance. While fully self-aware AI remains theoretical, ongoing research explores its ethical implications and technical feasibility, aiming to advance AI towards higher levels of self-awareness (Boucher, 2020).

***AI Applications***

Applications of AI include expert systems, ML, NLP, computer vision, speech, planning, and robotics.

ML is a branch of AI that focuses on creating systems that can learn from data and make decisions or predictions based on that data. ML can be applied to various domains, such as computer vision, natural language processing, recommender systems, and more. Additionally, Natural Language Processing (NLP) is a specialized field within AI that focuses on the interaction between computers and human languages. Its overarching goal is to equip computers with the ability to comprehend, analyze, generate, and manipulate natural language texts and speech. The applications of NLP span a wide range and include machine translation, speech recognition, sentiment analysis, information extraction, text summarization, question answering, and the creation of chatbots.

Expert systems are AI-driven computer programs designed to tackle complex issues within defined domains. They consist of a knowledge base, an inference engine, and a user interface. Drawing from facts and rules sourced from human experts or authoritative references, the inference engine employs logical reasoning to derive new conclusions. The user interface enables interaction, allowing users to ask questions, offer feedback, or update the knowledge base. Despite their utility, expert systems may falter when faced with unforeseen circumstances and typically lack common sense or inventive problem-solving capabilities.

Computer vision is an integral field within AI that empowers machines to comprehend and make decisions based on visual information, mirroring human vision capabilities. Some common computer vision algorithms include Sobel, Prewitt, or Canny for image filtering and edge detection, ANNs for image classification, object detection, and image segmentation, U-Net for image segmentation, SIFT (Scale-Invariant Feature Transform) and SURF (Speeded Up Robust Features) for feature extraction, Lucas-Kanade and Horn-Schunck for estimating the motion of objects in consecutive frames of a video, and Eigenface, Fisherfaces, and Local Binary Patterns (LBP) for face recognition (Szeliski, R., 2010).

Last but not least, AI Planning is a specialized field within AI dedicated to the development and execution of plans to achieve specific goals. The core tasks in AI Planning involve determining a sequence of actions that transition from an initial state to a desired goal state, followed by the execution of these actions in either a real or simulated environment.

## 4.3 Generative and Non-Generative AI

Generative AI (Gen-AI) and non-generative AI (Non-Gen AI) represent two broad categories of AI systems based on their capabilities and functionalities.

### *Gen-AI*

In January 2021, a breakthrough in AI creative abilities was announced when DALL-E was open for the public to experiment with. DALL-E is a platform that converts text to images through generative AI. In July 2022, another breakthrough in AI creative abilities was announced when MidJourney was open for the public to experiment with. MidJourney is another text-to-image generative AI model, but it was resulting in more realistic generated images with more options.

Since then, DALL-E and MidJourney were developed, and new versions were published gradually with more options including inpainting, outpainting, and image-to-image generation reaching DALL-E v3 and MidJourney 5 at the time of writing this research. Also, other image generation models were developed including Adobe Firefly and stable diffusion.

Of course, Many fields have leveraged the power of such models and capabilities and the architectural field is no exception and many applications of using image generation AI models in architecture were explored in section 7.2.2.

In fact, the advent of both generative and non-generative AI has significantly transformed the architectural design process. Generative AI, with its prowess in divergent thinking and algorithmic creativity, has become a catalyst for idea generation. Furthermore, generative AI facilitates collaboration between architects and machines, fostering a symbiotic relationship that leverages the strengths of both.

Gen AI refers to systems that have the ability to generate similar new content, often in the form of images, text, sound, 3d models, speech, code, video, etc. These systems can create outputs that are not explicitly present in their training data by understanding the distribution of data.

*Text Generation*

Text generation techniques encompass a diverse array of algorithms and models aimed at producing coherent and contextually relevant textual content across various domains. Markov Models leverage probability-based predictions, while Recurrent Neural Networks (RNNs) process sequences with hidden states to capture context. Long Short-Term Memory (LSTM) Networks overcome limitations in capturing long-range dependencies, and Generative Adversarial Networks (GANs) produce realistic text through an adversarial training process. Transformers, like BERT, utilize self-attention mechanisms to enhance contextual understanding, and Large Language Models (LLMs) represent a cornerstone in natural language processing (NLP), pre-trained on extensive text corpora. LLMs, such as GPT and BERT, demonstrate remarkable performance in tasks like text generation and classification. They are adaptable across various domains, from general language understanding to domain-specific applications like medical data analysis. With their pervasive applications in education, healthcare, customer service, and beyond, LLMs are shaping the future of human-computer interaction and information processing.

*Image Generation*

Image generation AI encompasses algorithms and models designed to create new images, leveraging deep learning architectures trained on large datasets. Key approaches include Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), PixelCNN/RNN, Deep Dream, CLIP, and DALL-E. GANs, like DCGAN and StyleGAN, produce diverse images, while VAEs capture latent representations. PixelCNN/RNN generate images pixel by pixel, Deep Dream enhances patterns, CLIP generates images from textual prompts, and DALL-E generates images based on textual descriptions by directly creating pixels in the generation process. It does not explicitly involve a separate noising and denoising mechanism. Stable diffusion algorithms, such as Diffusion with Denoising Priors (DDPM) and Noise-Contrastive Estimation (NCE), use controlled noise to transform images gradually, contributing to realistic and diverse image generation (figure 4-1).

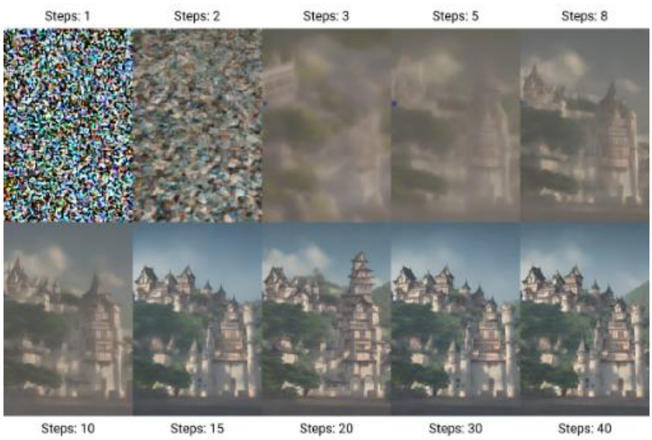



*Figure 4-1- The denoising process used by Stable Diffusion. https://en.wikipedia.org/wiki/Stable_Diffusion#/media/File:X-Y_plot_of_algorithmically-generated_AI_art_of_European-style_castle_in_Japan_demonstrating_DDIM_diffusion_steps.png*

Last but not least, inpainting and outpainting techniques, utilizing generative models, fill in missing or extend image content. These methods cater to various image generation requirements, highlighting the versatility and significance of AI in creating visually compelling content.

*Videos Generation*

Video generation in AI uses advanced techniques to create realistic and dynamic video content, simulating the appearance and motion of real-world videos. This field, part of generative modeling, leverages models like GANs, RNNs, LSTM networks, VAEs, 3D CNNs, and flow-based models to understand and synthesize dynamic visual content. GANs employ adversarial training to create realistic video sequences, capturing temporal and spatial patterns from video datasets. RNNs and LSTM networks handle temporal dependencies between frames, predicting the next frame to maintain coherent sequences. VAEs learn probabilistic distributions from training videos, sampling from these distributions to produce diverse video sequences. 3D CNNs process video data in three dimensions, capturing spatial and temporal features simultaneously from video volumes. Flow-based models focus on learning motion and transformations between frames, generating videos with realistic motion by modeling the underlying dynamics. These algorithms, trained on large video datasets, adjust model parameters to minimize the difference between generated and real videos, resulting in the creation of diverse and realistic visual content.

*3D Models Generation*

3D model generation in AI involves using advanced techniques to create three-dimensional representations of objects, scenes, or environments from 2D images, point clouds, or other input data, enabling AI systems to understand and replicate the three-dimensional nature of the physical world. Several notable models and algorithms are employed for this task, each leveraging different approaches. Depth Estimation Networks use CNNs to predict depth information from 2D images, learning from datasets containing RGB images and corresponding ground truth depth maps. Point Cloud Generative Models, including PointNet and PointNet++, generate

3D point clouds representing object surfaces, trained on datasets with 3D point cloud representations. Volumetric Representations with 3D CNNs divide 3D space into small cubes (voxels) and use CNNs to learn and generate volumetric representations from 2D images.

GANs are adapted for 3D model generation through adversarial training, refining the generator's ability to create realistic 3D structures from input data such as 3D models or point clouds. Multi-View Synthesis uses multiple 2D images from different viewpoints to synthesize a coherent 3D representation, leveraging neural rendering techniques and training on datasets of multi-view images.

Neural Radiance Fields (NeRF), introduced by Mildenhall, B., et al., 2020, represents 3D scenes as continuous functions outputting color and density for any 3D point, excelling at modeling complex scenes with detailed geometry and appearance. NeRF is trained on images captured from different viewpoints and uses ray marching to estimate color and density values, allowing for realistic and high-fidelity 3D representation. These algorithms are trained on diverse datasets containing 2D images, 3D point clouds, or volumetric representations, adjusting model parameters to minimize differences between generated 3D structures and real-world examples, thus enabling the reproduction of intricate three-dimensional scenes and objects.

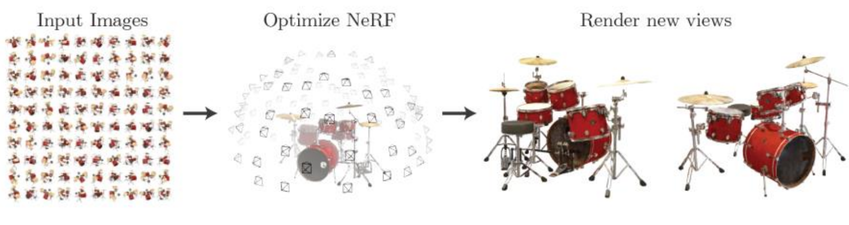



*Figure 4-2 – NeRF pipeline - Mildenhall, B., et. Al., 2020, NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*

The examples of algorithms mentioned in this section showcase the versatility of generative AI in creating content across visual art, music, text, and more. As the field continues to advance, generative models are likely to contribute to various creative and practical applications.

***Non-Gen-AI***

Non-generative AI focuses on tasks such as classification, regression, pattern recognition, and prediction, rather than generating new content. These systems are trained on labeled datasets to learn the relationship between features and labels. Applications include image classification with CNNs, where objects within images are identified and categorized, and NLP models for sentiment analysis, such as BERT or LSTM, which determine the sentiment of a given text. Regression models are widely used in finance for predicting stock prices based on historical data. Other applications of non-generative AI include clustering, association rules, dimensionality reduction, object detection, and face recognition. These models excel in prediction and pattern recognition, making them valuable tools across various domains.

In summary, generative AI focuses on creating new and original content, while non-generative AI is more oriented toward tasks that involve classification, regression, and pattern recognition without the explicit generation of new data. Both approaches have their own strengths and applications, and the choice between generative and non-generative AI depends on the specific requirements of the task at hand.

## 4.4 ML Definition and Types

Non-generative AI (ML) excels in analytical support, aiding architects in data analysis, simulation, and visualization. It contributes to realistic renderings, project management optimization, and quality assurance. However, the integration of AI in architecture poses challenges, including ethical considerations regarding biases, the delicate balance between technological efficiency and human creativity, and the need for architects to adapt to evolving workflows. In navigating these challenges, architects can harness the benefits of AI to enhance their design processes, ensuring a harmonious integration of technological advancements with traditional practices.

ML is a sub-set of AI which aids in discovering intricate patterns within data, utilizing good generalization on unseen data with very precise predictions. In ML several disciplines meet such as database, data mining, pattern recognitions, etc. This process does not require explicit programming of ML algorithms and meanwhile, many explorations in applying ML are being widely conducted in different fields. ML algorithms could be categorized as supervised learning, unsupervised learning, and reinforcement learning. Specific types of data are used with ML algorithms including images, text, numbers, and sounds. However, all these types require the ability to be transformed into numerical values so that they can be processed by machines. In architectural form design, numerical predictions require being labeled to be used in their predefined parameters afterwards to generate a model.

***Types of ML***

Machine learning can be broadly categorized into three main types based on the learning style and approach:

*Supervised Learning*

In supervised learning, the algorithm is trained on a labeled dataset, where each input data point is paired with its corresponding output or target. The goal is to learn mapping from inputs to outputs, enabling the algorithm to make predictions or classify new, unseen data. ML supervised learning algorithms perform both regression and classification tasks. Regression quantifies the relationship between input and target variables, predicting numerical values, suitable for predicting architectural parameters like lengths, widths, heights, and distances. Classification categorizes inputs into classes, useful for making binary decisions in architectural models, such as detecting the presence of windows in a wall. Artificial Neural Networks (ANNs), a subcategory of ML under deep learning, consist of interconnected nodes or perceptrons and have advanced significantly, enhancing capabilities in both regression and classification tasks.

*Unsupervised Learning*

Unsupervised learning deals with unlabeled data, exploring its inherent structure and patterns without explicit guidance to discover relationships, clusters, or representations within the data. Key techniques include clustering algorithms, which group similar data points; dimensionality reduction algorithms, which reduce feature numbers while preserving essential information; and association rules, which describe relationships among items in a dataset. Association rules are commonly used in market basket analysis to identify products frequently purchased together. An association rule is typically written in the form "If {X}, then {Y}," indicating the relationship between two itemsets. These techniques help uncover patterns, correlations, and co-occurrences in data.

*Reinforcement Learning*

Reinforcement learning involves an agent learning to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The agent, which is the decision-making entity, learns optimal strategies over time through its actions. The environment is the external system with which the agent interacts. Examples of applications include game playing, where agents learn through trial and error, and robotics, where robots learn to perform tasks in a physical environment.

These three types represent the fundamental paradigms in ml, each serving different purposes and applications. Additionally, there are hybrid approaches and specialized techniques within these categories, contributing to the diversity and richness of the field.

## 4.5 Data Sets in ML

In ML, a data set is a collection of data points that are used for training, testing, and evaluating machine learning models. Datasets play a crucial role in the development and assessment of ML algorithms, allowing models to learn patterns and make predictions based on the provided information. Data provided to algorithms could be in many forms including numerical, text, images, audio, videos, etc. according to the problem under study.

Any data set consists of features or input variables (attributes) which the model uses to make predictions. Inputs could be either numeric or categorical and this defines the problem and how it is approached. On the other hand, labels or outputs are the desired output or target values that the model aims to predict.

Data sets could be either labeled or unlabeled. While labeled data are data which contain input features with corresponding labels, unlabeled data contain only input features without corresponding labels. Labeled data is used to train supervised learning models for either regression or classification tasks. On the other hand, in unsupervised learning and with the absence of labels, the model’s task is to discover patterns or structures in the data without explicit guidance.

***Data set splitting***

Data set splitting is a crucial step in machine learning, dividing a dataset into training, validation, and testing subsets to assess model performance and prevent overfitting. The training dataset, comprising input-output pairs, is used to train the model. The testing dataset, separate from the training set, evaluates the model's generalization to new data. A validation dataset, distinct from both training and testing sets, helps tune hyperparameters and avoid overfitting. For small datasets, a split of 70-80% for training and 20-30% for testing is common. Large datasets allow for an additional validation set, with splits typically being 60-70% for training, 15-20% for validation, and 15-20% for testing. Random shuffling ensures representative subsets, and stratified splitting maintains class proportions in imbalanced datasets. Cross-validation, like k-fold, involves multiple splits for robust evaluation.

***Challenges associated with having an ML-ready data set***

Handling datasets in machine learning poses several challenges that can impede model generalization on unseen data. These challenges include data quality, imbalance, dataset size, dimensionality, noise, and missing values. Poor data quality can lead to biased models and inaccurate predictions, while imbalanced class distributions can result in biased models favoring majority classes. Noise and missing values further complicate model training, requiring careful preprocessing. High-dimensional spaces exacerbate the curse of dimensionality, necessitating techniques like feature selection and dimensionality reduction. Scaling and normalization address numerical feature discrepancies, while one-hot encoding handles categorical variables. Prior to model training, thorough exploratory data analysis and cleaning are essential to ensure dataset organization, balance, sufficiency, dimensionality, and cleanliness, as these factors significantly impact model performance (Aggarwal, C., 2015).

## 4.6 ML Algorithms

Machine learning algorithms are computational methods or procedures used by machines to learn from data and make predictions or decisions without being explicitly programmed. These algorithms enable machines to improve their performance on a specific task over time through the experience gained from the data they process. There are various types of machine learning algorithms, and they are explained in this section.

### *A) Regression*

Regression analysis quantifies relationships between a dependent variable (also known as the target variable) and independent variables (also known as features or covariates), aiding predictions or inferences. Ordinary Least Squares (OLS) is a common method, minimizing error sum of squares to establish a 'best fit' line. Assumptions include uncorrelated, zero-mean, and constant-variance errors for parameter estimation, with normal distribution for hypothesis tests and interval estimation. Advanced tests verify these assumptions for specific regression equations, making regression analysis a vital tool across scientific disciplines.

*Ensemble learning*

Ensemble learning combines decisions from multiple ML models to reduce error and enhance predictions compared to a single model. The maximum voting technique is then applied to aggregated decisions for the final prediction. In an ensemble of trees, each tree is grown based on a random vector realization, and final predictions are generated through voting with equal weights. Ensemble learning utilizes Bagging (parallel ensemble) and Boosting (sequential ensemble) methods. Bagging, introduced by Breiman in 1996, builds each tree using a bootstrap sample drawn with replacement from the training dataset. This reduces prediction errors by averaging and variance reduction, as explained by Breiman in 1998 and further detailed by Hastie et al. in 2001. Random forests further reduce variance by minimizing correlation between aggregated quantities. Ensemble learning algorithms include random forests, decision trees, XGBoost, etc.

A random forest regressor aim to reduce correlations by introducing an additional level of randomness. These models utilize a random subset of variables, selecting a subset of covariates at random. Cerquitelli, T., et al. (2019), investigated the use of random forest and ridge regression to forecast power consumption in buildings using the SPEC engine. Their study demonstrated the effectiveness of these methods in forecasting both fine-grained values and coarse levels of power consumption in buildings. Breiman asserts that random forests possess two key advantages: They achieve remarkable prediction accuracy. And this high accuracy is achieved across a broad spectrum of settings for the single tuning parameter utilized.

Additionally, Decision trees utilize a modified version of the C4.5 algorithm introduced by Quinlan in 1993. In regression trees, leaf nodes can represent distinct values corresponding to the concept or include a function for determining the value of the target attribute. Yu, Zhun, et al. (2010), employed a decision tree algorithm to predict the energy performance indexes of residential buildings, achieving a model accuracy of 92%.

Last but not least, XGBoost, short for eXtreme Gradient Boosting, utilizes additive modeling by sequentially incorporating new decision trees to minimize loss through gradient descent. This strategy prevents overfitting by integrating the outputs of existing trees with those of the new tree until loss is minimized or a predefined tree limit is reached. Yucong, W., and Bo. W., in 2020, introduced the EA-XGBoost model for predicting buildings' energy consumption, achieving an R2 score of 0.93, an MAE of 46.82, and an RMSE of 47.01.

*Ridge*

Ridge regression is a method used to improve model performance in the presence of multicollinearity within the data. By utilizing L2 regularization, it tackles scenarios where multicollinearity results in unbiased least-squares estimates and high variances, leading to substantial deviations between predicted and actual values.

*KNN*

The k Nearest Neighbor (k-NN) method, categorized as a non-parametric and supervised technique, requires three essential elements: a set of labeled entities, a metric for calculating object distances or similarities, and the specification of k, representing the number of nearest neighbors. In a study by Hong, G., et al. (2022), the k-NN regressor was examined for predicting hourly energy consumption in community buildings. The researchers concluded that the algorithm's overall RMSE results fell within the acceptable range according to ASHRAE guidelines.

*Linear Regression*

Linear regression is employed to assess the relationship between independent and dependent variables, aiming to find optimal coefficients (w = w1, ..., wn) that accurately represent a linear correlation. In a study by Boukarta, S. (2021), linear regression was examined to predict the annual energy demand for heating and cooling in residential buildings, despite the small sample size of only 60 samples. Remarkably, the model achieved an impressive R2 score of 0.94%.

*Multivariate Polynomial Regression*

Polynomial regression is a method that allows for flexible curve fitting, particularly focusing on a single independent variable X. When applied to situations with multiple independent variables, it is referred to as Multivariate Polynomial Regression. In a study conducted by Mavromatidis, L., et al. (2013), polynomial regression models were investigated for predicting the thermal performance of composite dynamic building envelopes. The results of the models indicated errors that were deemed acceptable under most conditions.

***B) Classification***

Classification is a data mining technique that is employed to predict the categorization of data instances into specific groups. A set of classification algorithms could be used to predict the classification of data including Random Forest, k-NN, Radius Neighbor Classifier, and MLP. The main difference between these algorithms when used in regression and when used in classification is that instead of analyzing and predicting continuous values, binary values are predicted.

***C) ANNs and Deep Learning***

ANNs constitute a broad class of machine learning models inspired by the human brain's functioning. They can be shallow or deep, with varying numbers of layers. Deep learning, a subset of machine learning, specifically focuses on neural networks with multiple hidden layers, allowing for the creation of deep neural networks. This characteristic depth enables deep learning models to automatically learn intricate hierarchical representations of data, enhancing performance in handling complex patterns. Deep learning algorithms include Multilayer Perceptron (MLP), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Transformer Models (e.g., GPT, BERT). MLP, a type of ANN, is particularly effective in regression tasks. It comprises artificial neurons, which take input values and apply weights specific to the neuron before passing through an activation function. Deep learning finds extensive applications in image and speech recognition, natural language processing, and strategic gaming due to its ability to learn hierarchical features automatically. Figure 4-3 shows the architecture of deep ANN.

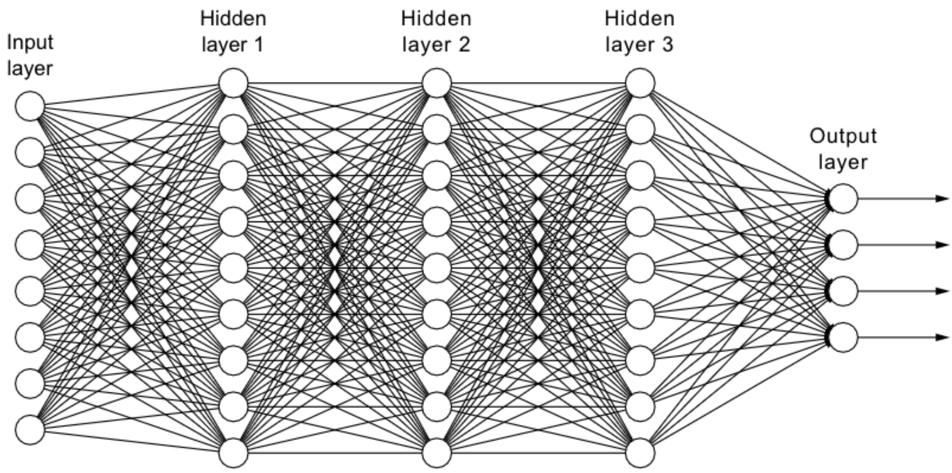



*Figure 4-3- Architecture of a deep ANN - https://freecontent.manning.com/neural-network-architectures/, Last Access: 30-8-2023*

## 4.7 A Review on Using Non-Gen-AI in Architecture

Non-Gen-AI applications include tasks like classification, prediction, pattern recognition, etc. This kind of application does not include creative generation of new outputs such as images, text, 3d models, etc.

Non-Gen-AI offers a diverse range of applications in architectural projects, spanning from project scheduling and cost estimation to energy performance analysis and building code compliance. Machine learning algorithms can optimize schedules, allocate resources, and predict delays, while also assisting in estimating construction costs and analyzing building designs for energy efficiency. AI tools can ensure compliance with building codes, aid in site selection, and enhance facility management through predictive maintenance and energy optimization. Additionally, collaborative design tools powered by AI facilitate real-time collaboration and intelligent design suggestions, while optimization algorithms help in material selection based on factors like cost and sustainability. Accessibility analysis further ensures compliance with accessibility standards, making AI a valuable asset in various facets of architectural design and management.

In 2023, Topuz, B., and Alp., N., reviewed the applications of ML in different architectural design sub-disciplines with themes including CAD, Computer-Aided-Engineering (CAE), and Computer-Aided Manufacturing (CAM). The researchers focused on 60 articles published in different journals. These studies spanned 21 different areas in architecture, addressing a myriad of challenges within each domain as shown in figure 4-4.

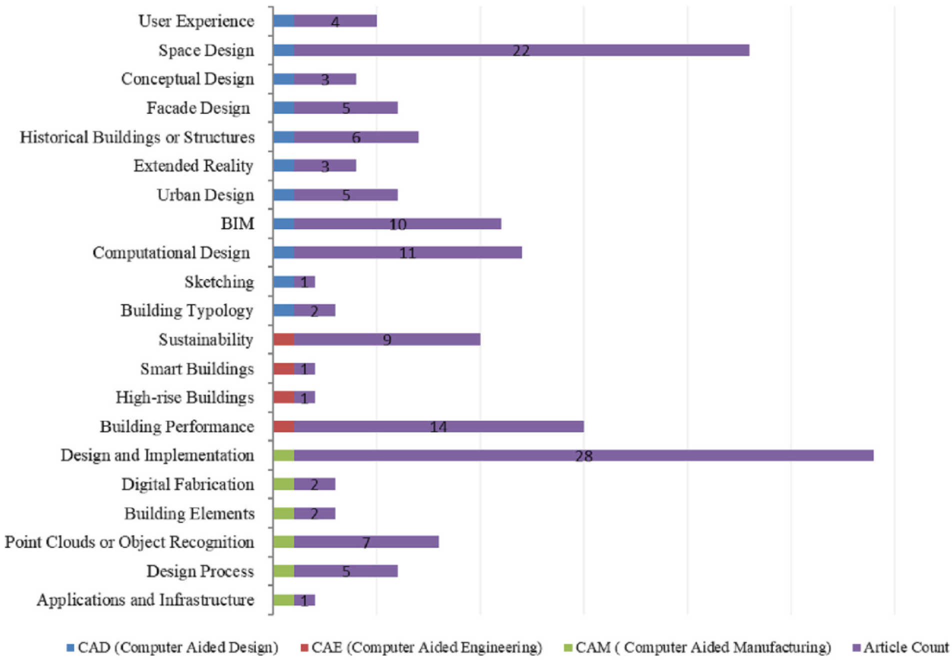



*Figure 4-4 Architecture subfields in 60 articles discussing applying ML in architecture (Topuz, B., and Alp, N., 2023, Machine Learning in Architecture)*

The breakdown exemplifies the extensive range of applications for machine learning in architecture, showcasing its potential to address problems related to design optimization, historic preservation, sustainability, building performance, and many other facets of the architectural process.

In the following discussion, different applications of non-gen-AI are presented.

<u>*Sustainability and Energy Efficiency*</u>

Numerous studies within sustainable architecture have employed Machine Learning (ML) techniques to optimize building energy efficiency and consumption. For instance, Tansas and Xifara developed a statistical ML framework to scrutinize the influence of variables like wall area and glazing area on residential building heating and cooling loads, emphasizing the accuracy of ML predictions aligned with the training data. Chou and Bui, 2014 utilized various AI techniques to estimate heating and cooling loads, with the ensemble approach and Support Vector Regression standing out. Robinson et al., 2017 found that gradient boosting regression models excelled in predicting commercial building energy consumption.

Also, Roy et al., 2023, explored advanced ML techniques for residential buildings, while Deng et al., 2018 cautioned about nuanced performance compared to linear regression for US commercial building energy use. Studies by Rahman and Smith showcased ML's capability, including Neural Networks and Gaussian process regression, in predicting fuel consumption in commercial buildings. Additionally, Fan et al. applied Deep Learning for short-term building cooling loads, demonstrating DL's potential for accurate prediction models. Yang et al. introduced an adaptive artificial neural network capable of predicting unexpected data behavior. Gonzalez and Zamarreno employed a feedback ANN for short-term electric load consumption prediction, highlighting its simplicity and resource efficiency.

Moreover, Kristianse, T., et al., 2022, explored applying Artificial Neural Networks (ANNs) to predict annual daylight illuminance and operative temperature, aiming to reduce simulation time, achieving a promising 96% reduction in overall time using ANN models. These studies underscore the versatility and effectiveness of AI and ML techniques in addressing various challenges associated with electricity consumption prediction in diverse building contexts. Additionally, Sebestyen, A., 2020, evaluated ML model predictions of radiation values and sunlight hours compared to software plugin metrics like Ladybug toolsets in Grasshopper3d for Rhinoceros3d. In 2019, Feng et al. introduced a method blending parametric design with ML algorithms to evaluate early-stage environmental performance in building design, addressing uncertainties associated with design choices. Singh, M., et al., 2022, developed a convolutional neural network approach for energy prediction, addressing key challenges and providing interpretable BPS information through a web tool. He, Y., et al., 2021, introduced a hybrid framework for rapid evaluation of pedestrian-level wind environments in architectural sustainable design, demonstrating efficiency and accuracy in providing design optimization information. Collectively, these studies highlight ML's potential to enhance sustainability and efficiency in architectural design and energy consumption prediction.

*Architectural Theories*

Several studies have leveraged machine learning (ML) techniques to enhance various aspects of architectural analysis and design. Uzun and Colokoglu (2019) utilized a pretrained Faster-RCNN-Inception-V2-COCO model to classify architectural drawings images into plans and sections, highlighting challenges due to dataset size. Wang et al. (2022) employed CNN-based models for classifying architectural styles, achieving satisfactory performance with preprocessing and attention mechanisms. Xu et al. (2014) addressed the multi-class problem in architectural style classification using probabilistic analysis and deformable part-based models. Sun et al. (2022) proposed a deep learning framework for understanding architectural styles and age epochs, demonstrating its effectiveness in analyzing building façades.

Moreover, Shalunts et al. (2011) successfully clustered façade elements by architectural style using k-means, achieving high accuracy percentages ranging between 92.5 and 98.1 according to the class. Alymani et al. (2019) introduced a workflow combining database systems and unsupervised learning algorithms to cluster architectural design aspects, highlighting the efficacy of K-Means clustering. Millan et al. (2022) presented a methodology utilizing data analysis and machine learning to track the design process in architecture, revealing insights into design strategies and problem-solving pathways. Qin et al. (2023) introduced the "NeoDescriber" model for automatic identification and description of Neoclassical buildings, achieving effective performance in classification and detection tasks. These studies collectively demonstrate the diverse applications and promising outcomes of ML in architectural analysis and design.

*Form Prediction*

Cudzik and Radziszewski (2018) trained an artificial neural network (ANN) to predict detailed configurations of Roman Corinthian order capitals, positioning the algorithm as a co-designer capable of generating potential spatial variations for examined forms. The dataset included samples enabling the analysis of local deformation, with input data comprising sample coordinate values, surface normal vectors, and volume center plane deviations. Through the backpropagation of errors learning procedure, the ANN was trained with 28,900 samples, resulting in a Mean Square Error below 0.001. The successfully trained ANN demonstrated its ability to generate three-dimensional variations of new capital forms based on given input parameters, enhancing the design process by providing computer-generated solutions. This research showcases the valuable role of neural networks in architectural computational design, extending the range of available design tools.

*Figure 4-5 - Designed capitals with ML - Cudzik, J., 2018, Artificial Intelligence Aided Architectural Design*

Lakzaeian (2020) addressed challenges in multi-planar building facades segmentation by introducing a specialized algorithm designed to differentiate between structural and non-structural elements in complex facades. The algorithm achieved a 98% accuracy for single complex openings and an overall average accuracy of at least 91% when applied to buildings in Dublin, Ireland. Additionally, Zheng and Yuan (2021) developed a specialized artificial neural network aimed at enhancing the precision and computational efficiency of learning and generating 3D geometries as vectorized models. This involved creating a custom data structure with feature parameters aligned with the neural network's requirements, resulting in improved design feature extraction.

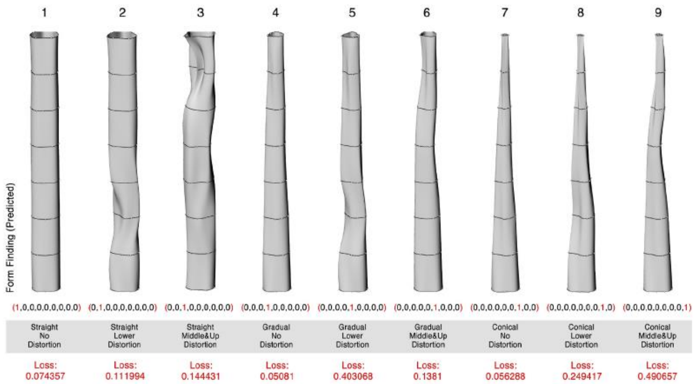



*Figure 4-6 - Expected forms and predicted forms from test dataset - Zheng, H., and Yuan, P., 2021, A generative architectural and urban design method through artificial*

The neural network's generative abilities demonstrate its predictive power, efficiently learning and extrapolating geometric design features from existing building data. This data-driven approach provides designers with a powerful tool for informed and efficient design exploration.

*BIM Models Semantics*

In their 2018 study, Bloch, T., and Sacks, R., explored the classification of room types in residential apartments using an ANN algorithm, comparing it to rule-inferencing. The research highlighted the direct applicability of machine learning to space classification, while rule-inferencing proved unsuitable for this context. This underscores the importance of selecting appropriate AI methods for specific BIM object classification challenges. Similarly, in 2019, Koo, B., et al. utilized support vector machines (SVM) to evaluate the semantic integrity of mappings between BIM elements and IFC classes. Their approach, trained on a dataset of 4187 unique elements from six architectural BIM models, demonstrated high accuracy in classifying elements and subtypes within classes. These studies contribute to automating quality checks in BIM deliverables and enhancing semantic enrichment for domain-specific analysis.

*Design and Fabrication*

In 2020, Yazici, S. integrated ML and ANN algorithms with geometry, material, and structural performance simulation data to support decision-making processes. They trained an ANN, non-linear regression model (NLR), and a Gaussian mixture model (GM) using data from structural performance simulations to predict materials based on architectural geometry and panel clusters on the shell model. The results demonstrated fast solutions and accurate predictions, offering valuable insights for decision-making.

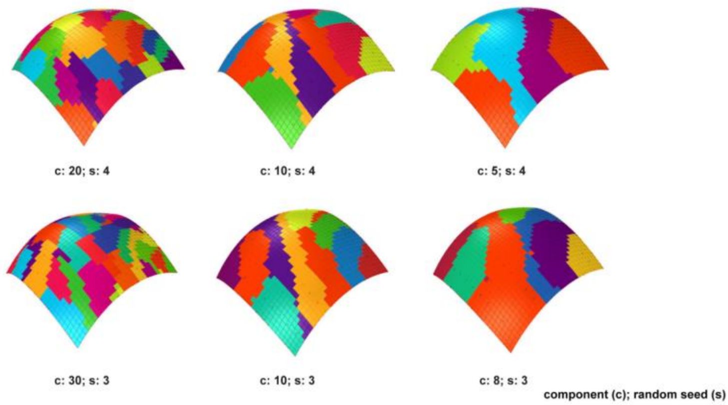



*Figure 4-7 - GM algorithm implemented towards prediction of panel clusters based on the area size and planarity of panels. - Yazici, C., 2020, A machine-learning model driven by geometry, material and structural performance data in architectural design process*

In 2018, Tamke, M., et al. explored ML's application to enhance design and fabrication adaptation in Robotic Incremental Sheet Forming (RISF). ML was used to manage forming tolerances by creating, adapting, and improving fabrication instructions. The integration of ML into fabrication processes utilized data mining techniques and trained ML models on physical outputs, acquired via 3D scanning of ten panels, yielding approximately 45,000 samples. Two approaches were employed: a regression-based method for local adjustments within panels and a neural-network-based approach for predicting and adjusting entire panel geometries. The regression-based method used in-process measurements for achieving required tolerances, while the neural network predicted the final shape of the panel after fabrication, enabling adjustments based on predicted springback.

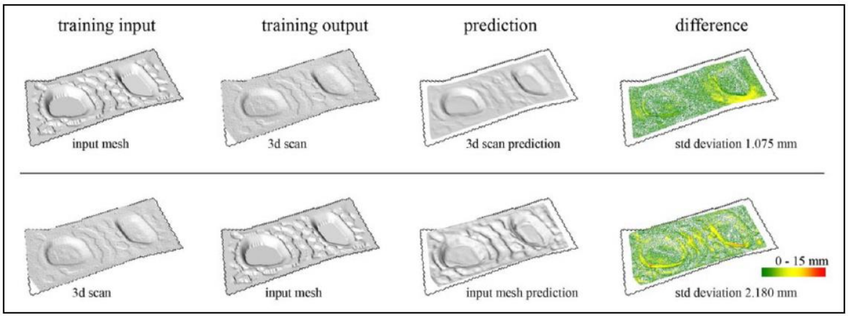



*Figure 4-8 - different input–output training sets and the achieved accuracy – Tamke, M., et. Al, 2018, Machine learning for architectural design: Practices and infrastructure*

## Layouts Design Evaluation

In 2020, Mandow, L., et al. explored sketch generation for energy-efficient single-family dwellings using a combination of shape grammars and reinforcement learning. Their approach involved defining shape grammar rules and applying reinforcement learning to generate habitable and energy-efficient sketches. The study highlighted the reinforcement learning process and provided experimental results demonstrating convergence, along with validation using energy simulation software. Moreover, in 2017, Takizawa, A., and Furtura, A., investigated spatial feature assessment using computer-generated modeling, VR, and deep learning. They utilized a computer-generated model of a street in Osaka to capture omnidirectional images, incorporating depth information at 50 observation points. Virtual reality preference evaluations informed the training of deep convolutional neural networks (DCNNs), revealing that the model error rate was significantly lower for RGBD images as well as the importance of integrating color/texture and geometric features for enhanced spatial evaluations. This interdisciplinary approach offers avenues for more accurate spatial analyses in urban environments, although further foundational research is needed for widespread applications.

## 4.8 A Review on Using Gen-AI in Architecture

As discussed in section 4.3, Gen-AI applications involve generating new content, ideas, or solutions based on input data or predefined parameters. In the context of architectural design, generative AI can play a significant role in creating, modifying, or optimizing designs by leveraging algorithms and computational models. In this section, some gen-AI applications in architectural design used by researchers and architects today are reviewed.

*A) Text Generation*

Text generation AI models could be used in various fields of architecture. In 2023, Caliskan, E., explored the potential applications of ChatGPT in third-year architectural design studios. The research involved structuring interviews with ChatGPT, with findings evaluated using the Delphi technique among experts. ChatGPT demonstrated the ability to address design issues but faced limitations in accessing maps and discerning geopolitical entities. Beyond documented studies, architects increasingly utilize large language models (LLMs) like ChatGPT, LlaMa, and Bard. These models aid in generating prompts for image generation tasks, conceptual design images, floor plans, and architectural details. Moreover, the human-like text generated by LLMs facilitates direct communication with clients and stakeholders, enabling architects to convey messages effectively.

In an experiment utilizing ChatGPT v3.5, architects explored its role in brainstorming architectural design concepts for a Mercedes-Benz exhibition. Appendix D showcases the conversation, revealing ChatGPT's organized responses and ability to ask pertinent questions, guiding the discussion. The model demonstrated awareness of project criteria, suggesting ideas aligned with Mercedes-Benz's precision and craftsmanship ethos. Furthermore, ChatGPT's responses became more specific as discussions progressed, highlighting its potential in generating innovative design concepts through iterative questioning and refinement. This process underscores the value of LLMs in enhancing brainstorming phases, fostering creativity, and unlocking new design possibilities.

*B) Generating Layouts*

Image generation AI in architecture has become an innovative and influential tool, providing architects and designers with new ways to conceptualize, visualize, and iterate on design ideas. There are several applications through which image generation AI started to influence the architectural field.

In 2019, Chaillous, S., utilized nested GANs to generate a diverse array of floor plan designs, employing a classification methodology for exploration. These nested GANs enabled the capture of complexity in floor plans and addressed challenges sequentially. Following this, a pipeline was employed to produce finalized plans with walls and furniture. Utilizing Boston's building footprints database, an algorithm was trained to generate footprints based on the layouts. Subsequently, algorithms were trained with over 700 annotated floor plans, each designated for a specific room count. Lastly, a model was trained to furnish entire units and expanded to include room-specific furnishing based on function (figure 4-9).


*Figure 4-9 - Resulting Furnished Units (Chaillous, S, 2019., AI & Architecture – An Experimental Perspective – Harvard University GSD)*

The model was further developed to solve entire buildings, adding windows and doors rationally. And the author applied a transfer-style method to train the GAN to create plans based on a specific architectural style as shown in figure (4-10).

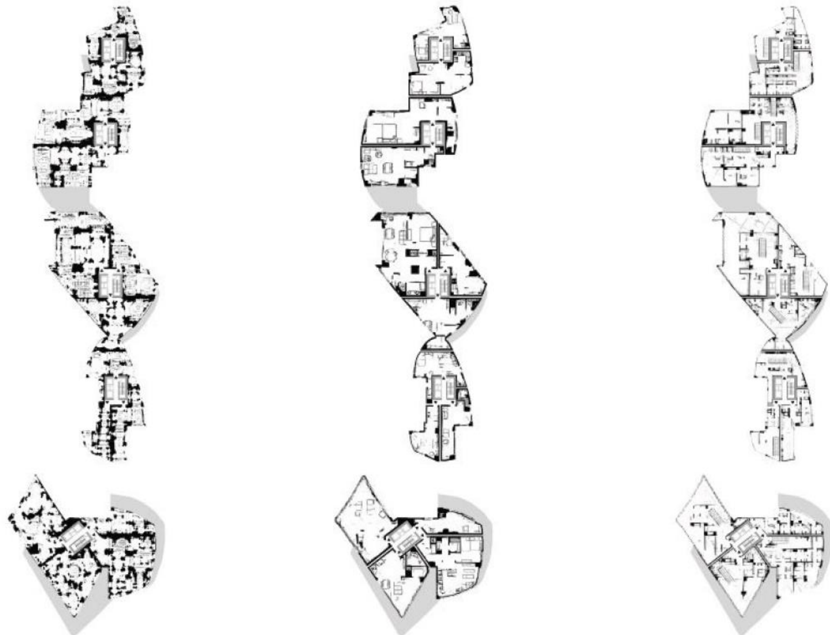


*Figure 4-10- 15th Floor Processed Under Each Style: Baroque (Far-Left), Manhattan (Center-Left), Row-House (Center-Right), Victorian (Far-Right) (Chaillous, S, 2019., AI & Architecture – An Experimental Perspective – Harvard University GSD)*

Afterwards, the classification algorithm categorizes the generated floor plans based on various criteria, aiding users in exploring different designs. This user-oriented approach allows architects to intervene in the pipeline's steps, making the process architect-centered.

Aalaei et al. (2023) explored architectural layout generation using graph-constrained conditional GANs, introducing methods for translating high-level constraints like bubble diagrams and implementing a fully vectorized workflow. Their key contribution involved applying a convolutional message passing (CMP) approach, considering both topological and geometric conditions. They presented a distinct network architecture and an iterative pipeline utilizing three separate GAN models with unique objectives. Figure 4-11 illustrates the proposed pipeline which includes user input, model-generated layout, user modifications, and final architectural plan and 3D model.


*Figure 4-11 - Iterative and collaborative human–machine workflow for architectural floor plan generation. (Aalaei, M., et. Al., 2023, Architectural layout generation using a graph-constrained conditional Generative Adversarial Network (GAN))*

Karadag et al. (2022) trained a Pix2Pix GAN algorithm using two datasets of educational school buildings' space layouts. They developed two algorithms: one generated footprints and suggested furniture layouts in block zoning, while the other generated furniture drawings in the plans. This innovative approach targeted the problem directly, instead of relying on existing datasets. The model successfully generated outputs not only from the training dataset but also from the validation dataset. Figure 4-12 displays the results of the trained model on the validation set.

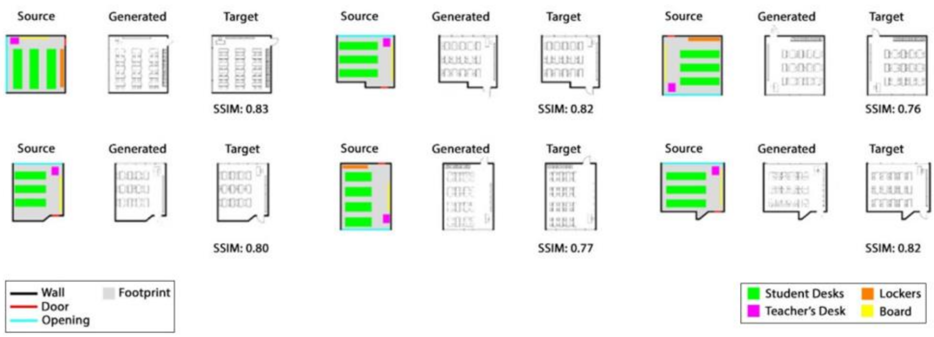



*Figure 4-12 - Test results on the validation data set of EDU-AI (Karadag, I., et. Al., 2022, EDU-AI: a twofold machine learning model to support classroom layout generation)*

As, I., et. Al, 2018, presented a deep Neural Network (DNN) approach utilizing graphs for the generation of conceptual designs. The system demonstrated its capability to assess and score designs, decompose them into fundamental building blocks (figure 4-13), and creatively recombine them into novel compositions. Additionally, a Generative Adversarial Network (GAN) method was introduced, capable of producing new designs which were not present in the training set.

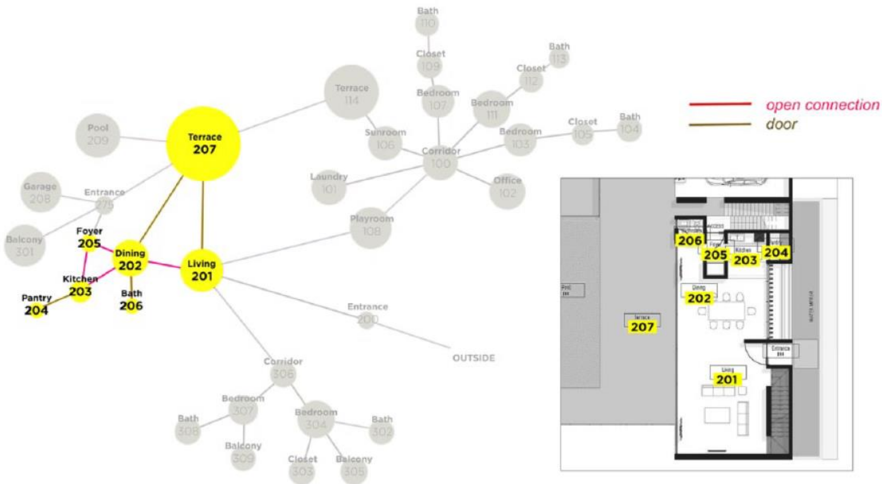



*Figure 4-13 - A subgraph, that is, building block, discovered by the DNN highlighted within the larger graph of a home (As., I., et. Al, 2023, Artificial intelligence in architecture: Generating conceptual design via deep learning)*

Also, Liu et al. (2022) utilized the Pix2Pix GAN algorithm to generate private garden layout plans based on given site conditions, learning from traditional Chinese private gardens.

*C) Modifying Images Contextually*

Moreover, Sun et al. (2022) employed GANs to abstract historic architecture styles and automatically generate stylized facades (figure 4-14). Their study curated a bespoke dataset from Harbin Central Street, implementing a data augmentation process. The generated designs were quantitatively and qualitatively assessed, demonstrating high accuracy, realism, and diversity. Two applications validated the feasibility and adaptability of the proposed workflow, enhancing historic urban area renovation design processes.

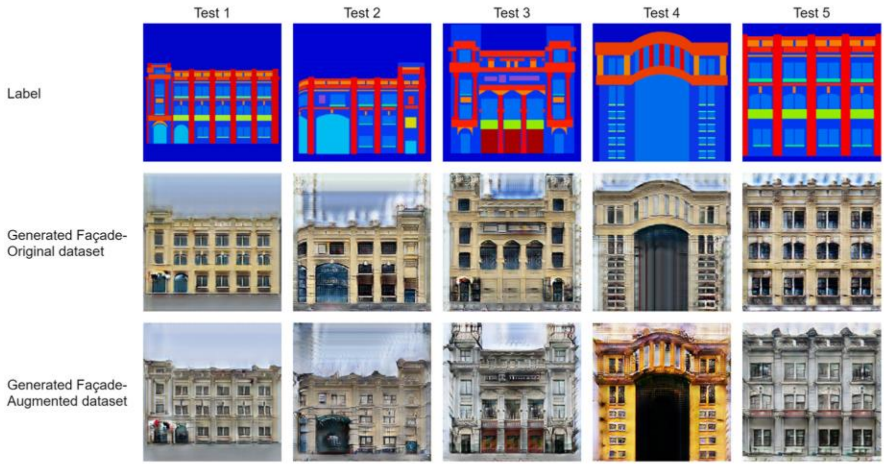



*Figure 4-14 - Generated facades for the Harbin Central Steer (Sun, C., et. Al., 2022, Automatic generation of architecture facade for historical urban renovation using generative adversarial network)*

## *D) Generating Images from Texts*

Basarir and Erol (2021) proposed an AI framework to generate architectural sketches based on client briefs, using semantic analysis and visual pattern recognition. Additionally, Ploennings and Berger (2023) explored the use of AI art platforms like Midjourney, DALL·E 2, and Stable Diffusion in concept design, noting their effectiveness in ideation, sketching, and modeling. They evaluated AI models' versatility in architectural tasks, highlighting successes in generating inspirational images and addressing challenges in responding to generic requests. The study also analyzed 85 million MidJourney queries, revealing prevalent usage patterns and suggesting structured workflows for interior and exterior design (figure 4-15). These findings not only showcased the current capabilities of image generation models in architectural design tasks but also illuminated potential advancements and avenues for further exploration in the integration of AI tools within the creative processes of architecture.


*Figure 4-15 – (left) Minimal workflow for Midjourney (a–d), DALL· E 2 (e–h), and Stable Diffusion (i–l) (Ploennings, J., and Berger, M., 2023, AI in Architecture), (right) Refinement and variant generation in Midjourney (a–c), DALL· E 2 (d–f), and Stable Diffusion for a walkway (g) and a second story (h, i) - (Ploennings, J., and Berger, M., 2023, AI in Architecture)*

## *E) Generating Images from Images and Text*

Bao and Xiang (2023) examined Stable Diffusion, MidJourney, and DALL-E 2 as smart assistants in preliminary design processes. They analyzed the impact of AI activities on architects and students using a survey with AI-generated images (figures 4-16 and 4-17). Results showed AI's potential to optimize architectural design by reducing time and enhancing visualization, with satisfactory performance reported by users.

*Figure 4-16 - Base input sketch for AI generation (Bao, Y and Xiang, C., 2023 - Exploration of Conceptual Design Generation based on the Deep Learning Model-Discussing the Application of AI Generator to the Preliminary Architectural Design Process)*


*Figure 4-17- Rendering generation results made by MidJourney, Stable Diffusion and DALL-E 2 (from left to right respectively). (Bao, Y and Xiang, C., 2023 - Exploration of Conceptual Design Generation based on the Deep Learning Model-Discussing the Application of AI Generator to the Preliminary Architectural Design Process)*

Hu et al. (2021) introduced the Low-Rank Adaptation (LoRA) method, optimizing large pre-trained language models for downstream tasks by decomposing weight updates (ΔW) and reducing trainable parameters. During inference, the weight update is seamlessly merged into the main weights without additional overhead, facilitated by a LoRA scaling factor (α). Kuang et al. (2023) proposed a workflow using LoRA to generate facade images of historical styles for urban renewal projects, preserving the city's historical identity. Utilizing the LoRA and ControlNet models, architects could automatically generate facade images of specific historical styles. This approach efficiently preserved and integrated historical architectural elements into urban renewal projects, contributing to the maintenance of the city's historical identity.


*Figure 4-18- Arcade facade renewal based on prompt and ControlNet. - Kuang, Z., et. Al, 2023, Advancing Urban Renewal: An Automated Approach to Generating Historical Arcade Facades with Stable Diffusion Models*

Although image generation using AI seems very promising in terms of visual appearance, ideas, and even consistency, some questions and concerns are present within this approach. These concerns will be discussed later in this chapter.

*F) 3d-Models Generation*

AI-driven methods, particularly Generative Adversarial Networks (GANs), are revolutionizing 3D model generation by autonomously producing intricate and realistic structures. Akizuki et al. (2020) applied

Reinforcement Learning (RL) within a 3DGAN framework to generate furniture 3D models with topological consistency by voxelizing thousands of furniture 3d models to train the algorithm, showcasing the algorithm's ability to create complex structures within specified constraints.

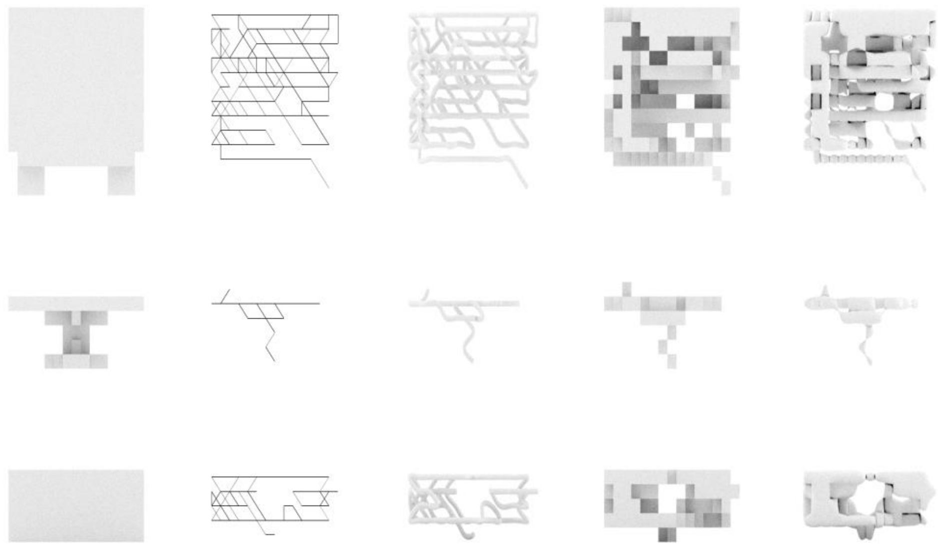


*Figure 4-19 - Object generation via 26-actions model - from left to right, input voxel geometries, voxels, pipes, and smoothened meshes. Akizuko, Y., et. Al., 2020, Generative Modeling with Design Constraints – Reinforcement Learning for Object Generation*

Nichol et al. (2022) utilized a two-step approach for 3D scene generation, employing text-to-image diffusion models (GLIDE) to create synthetic views, followed by generating 3D point clouds conditioned on the generated images. The models were trained on millions of 3D models, with diverse post-processing steps implemented to ensure data quality. To standardize the data, each 3D model was rendered from multiple angles as RGBAD images using Blender, ensuring consistency for subsequent analysis and processing.

*<u>Integrating AI to the Conventional 3D Modeling Pipelines</u>*

Liu et al. (2021) utilized style transfer NN algorithms to generate 2D photos from floor plans and truss structures, aiding in 3D modeling for architectural design. The algorithms processed floor plans as content images and truss structures as style images, generating transferred images.

These images were then translated into floor plans for innovative structure design, with vertical components serving as pillars. This transformation from 2D-style transferred images to 3D geometries presents new possibilities for architectural design processes.

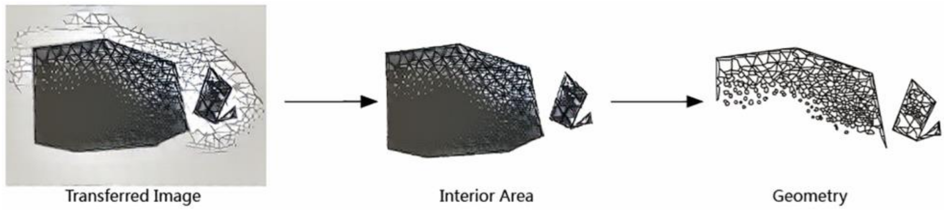



*Figure 4-20 - Transferred Image Translation (Liu, C., et. Al., 2021, Pipes of AI – Machine Learning Assisted 3D Modeling Design)*

AI generative algorithms allow other types of generation including codes and videos. In 2023, Ceylan et al. explored text-guided video editing with the 'Pix2Video' algorithm, utilizing pre-trained image models to achieve desired edits while preserving source video content. The method involves employing a pre-trained structure-guided model for text-guided edits on an anchor frame and propagating changes to future frames through self-attention feature injection.

Similarly, Chai et al. introduced 'StableVideo,' enhancing text-driven diffusion models to generate consistent appearances for edited objects, employing an inter-frame propagation mechanism within the diffusion video editing framework. Blattmann et al. applied the Latent Diffusion Model (LDM) paradigm to high-resolution video generation, fine-tuning on encoded image sequences and aligning diffusion model upsamplers for temporal consistency.

Moreover, Karras and colleagues developed DreamPose, a diffusion-based method for generating animated fashion videos from still images, achieving state-of-the-art results in fashion video animation by transforming a pre-trained text-to-image model into a pose-and-image guided video synthesis model. These advancements highlight the growing potential of AI-driven techniques in text-to-video and image-to-video generation for diverse applications.

To conclude, The integration of machine learning into the architectural design process holds immense potential across all planning phases, transforming both repeatable and predictable activities. Machine learning tools can effectively replace certain tasks, particularly those involving decision-making, by learning from the work performed by architects. This evolution introduces complex machine learning methods, bringing artificial intelligence to the forefront of architectural and product design. This shift has the power to redefine the value of algorithmic design, moving beyond being a mere computational tool to becoming an equal collaborator in the design process. This collaborative synergy between human designers and machine learning systems has the capacity to revolutionize the architectural design landscape, fostering efficiency, creativity, and the exploration of novel design paradigms.

Also, and more importantly, such collaboration could be described as 'human-centered' where the machine aids in automating the process rather than interfering in creative and aesthetic aspects of the design which we argue that they are the essence of an architectural product. These aspects along with many others require the architect to be the center of the process seeing the whole picture and taking decisions that respect the complicated network of all design aspects.

Recently, many AI applications have been introduced to the architectural design process including encompassing modeling, classification, rendering, and more. However, getting predictions that aid in form modeling was not experimented with deeply. Also, the basic knowledge of a framework to codify a building to retrieve its parameters and to create efficient data sets remains crucial for the success of such applications.

## 4.9 Generative AI Drawbacks in Architectural Design

Today, the field of architecture has seen many experiments with generated designs in the form of images. Recently, some architects have started to generate design ideas through image generation gen-AI models either by providing a prompt expressing the project requirements, some certain ideas, etc. in the form of text, or by providing sketches.

Gen-AI's impact on the authenticity of designs is a pivotal aspect of this evolution. The authenticity of generative AI architectural designs is a nuanced consideration, encompassing both the innovative potential of AI-generated creations and the preservation of unique human expression in design. Generative AI excels at exploring diverse design possibilities, pushing the boundaries of conventional architecture, and fostering creativity. However, questions arise regarding the authenticity of designs when algorithms autonomously generate solutions. Critics argue that reliance on generative AI might lead to a homogenization of designs, with the risk of overlooking the distinct cultural, historical, and contextual nuances that human architects often incorporate into their work.

Millet et. Al, 2023, revealed an anthropocentric bias in art appreciation, suggesting a prevailing human-centric viewpoint in assessing creativity, particularly in the context of AI-generated art. Their experiments, encompassing over 1,700 participants, unveiled a consistent bias against AI-created art, perceived as less creative and awe-inspiring compared to human-made counterparts. Similarly, Ragot et. Al., 2020's extensive study involving 565 participants identified a preference bias toward human-made creations, with art perceived as AI-generated receiving less favorable evaluations. These findings underscore a persistent negative perception bias towards AI in the realm of art, reflecting a broader inclination to view creativity as an exclusively human trait. As AI continues to advance in the creative domain, these biases pose challenges to fostering an inclusive perspective that recognizes and appreciates the unique contributions of both human and machine creativity. Addressing these biases is essential for cultivating a more open-minded appreciation of AI's potential as a tool for artistic expression.

Yet when it comes to the originality of ideas in image generation models like diffusion models, it is important to note that these models are trained on existing data. The originality of generated samples depends on the diversity and complexity of the training data. If the training data includes a wide range of unique and novel examples, the model has the potential to generate original outputs. However, diffusion models, like other generative models, do not inherently generate truly novel ideas in the creative sense—

they synthesize new examples based on patterns learned from the training data.

Moreover, gen-AI models could be eclectic. The term "eclectic" in the context of image generation could refer to the ability of a model to combine diverse elements from its training data to create novel and varied images. If a diffusion model has been trained on a diverse dataset containing images with different visual styles, objects, and scenes, it may be capable of generating images that incorporate elements from various sources.

In the case of diffusion models, the process typically involves iteratively adding noise to an input until it transforms into a sample from the target distribution. The ability to create eclectic images could arise from the model's capacity to blend and remix features it has learned from disparate examples in its training data (Rombach, R., et. Al., 2022).

Thus, proponents of generative AI emphasize its capacity to reinterpret and combine design elements in novel ways, challenging traditional notions of authenticity. AI-generated designs can be seen as a reflection of the data they are trained on, capturing and reinterpreting architectural styles and features from various sources. This dynamic process can result in unexpected designs that embody a new form of randomness rooted in computational creativity.

Navigating the authenticity of generative AI architectural designs requires a careful balance. Architects and designers must actively engage with AI tools, guiding the algorithms to align with their vision while also embracing the serendipity and novelty that AI can introduce. The synthesis of human insight and machine-generated possibilities can lead to truly authentic designs that are both innovative and deeply connected to human sensibilities. As the field continues to evolve, a thoughtful and critical approach to the integration of generative AI will be essential in preserving and redefining the authenticity of architectural design.

Among many researchers and architects, Chaillou, S., who utilized Gen-AI in creating floor plans in 2019 had beliefs which are rooted in the assertion that a statistical approach to design conception profoundly shapes the potential of AI in the field of architecture. The departure from deterministic methodologies toward a more holistic, less-prescriptive character is seen as a unique opportunity within the architectural domain. Rather than viewing machines solely as tools for optimizing predefined variables, Chaillou advocates relying on AI to extract significant qualities and emulate them throughout the entire design process, marking a paradigm shift toward a more dynamic and exploratory design experience.

Furthermore, according to Chaillou, the conviction lies in the pivotal role of designing the right pipeline to ensure the success of AI as a new architectural toolset. The preference for the "Grayboxing" approach, as introduced by Witt, A., 2018, is considered strategic and likely to yield optimal results. Chaillou contrasts this with the "black box" model, where users input information upfront and receive finished design options at the end, without influence over intermediate generation steps. The "Grayboxing" approach, as advocated by Chaillou, involves breaking down the pipeline into discrete steps, empowering the user to intervene at various stages as mentioned in chapter 1.

This hands-on control over the machine ensures the user's ultimate guarantee of the quality of the design process, offering a more collaborative and iterative interaction between human insight and AI capabilities. This deliberate approach, as expressed by Chaillou, underscores a commitment to a thoughtful integration of AI into architectural practices, emphasizing user agency and creativity within the technological framework.

The gray box approach seems logical especially with today's mathematical applications in architectural generative designs which includes optimization and simulation techniques for instance. Also, AI applications in architectural design could involve such an approach. Especially, that it includes different applications (APIs) that could be learnt and used in the form of an internal black box operation in the design process without needing to learn what is behind -as users and not as developers-.

In fact, while strongly agreeing with Chaillou that AI should be dealt with as a 'toolset' for the architect that involve many advantages, I strongly believe that today’s Gen-AI applications which generates images (used as designs) -at least till the time of writing this research- are as far as possible not only from what architectural design profession is about but also from what an architectural design methodology could be and could propose as a solution to a problem. I believe that an architectural design product is not just drawings. It is rather an experience and a process. And this process most likely -if not always- includes problem solving of a handful of issues from a handful of other disciplines as well as architectural rules (form, commodity, and delight). Those other disciplines include structural, societal, psychological, philosophical, humanitarian, and environmental, to name a few. All of these issues could never be diminished to whatever a generated architectural drawing image could encompass because every project should be designed with a whole new character and new thoughts.

Also, and more importantly regarding text-to-image applications specifically, ‘can all the aspects of an architectural design in words be diminished?’. Architects who mostly follow the ‘black box’ approach find it hard to clearly express their ideas and how the form is generated. And worse, even those who apply the ‘glass box’ approach either by following function, relying on generative design various techniques, etc., still have hard time realizing the process and the reasons of the resulting product which is usually hard to explain to a machine. A gen-AI photo generation model cannot understand the orientation of the building, or the parametric approach taken to stabilize the structure, or the best façade pattern or form manipulation to reduce solar gain. Instead, Gen-AI models generate responses based on patterns learned during training. And of course, such manners are taken into consideration from day 1 in the design process and are ‘applied directly’ more than ‘thought of’. If architects skip such techniques in the process and start with a generative design (created by AI-Gen models), it is most likely that the end product will be as far as possible from these images, and then the architects should ask themselves, ‘what was the benefit?’ Even with newly introduced models including LoRA and ControlNet. Still, the ‘control’ they provide the architect with, is more control of an outline or a boundary of the building or getting closer results to the words descriptions. Still, this whole process deals with an architectural product as a 2-dimensional product.

In addition, architectural design is about understanding what a user desires, and not only the architect's aesthetic and creative parts. If this part of the architect's job is well perceived, they would most probably find themselves in need of designing something that is unseen before, even if some details/techniques are reused. In this sense, relying on a dataset of previously designed projects could contradict this theory.

And in this manner, I would strongly suggest differentiating between the product of Gen-AI models and the product of a generative design as there are no contradictions if the previous theory on generative design is applied. As discussed in chapter 4, generative design is still controlled by the architect who defines the parameters and the goals for which a machine searches the solutions to achieve. Even when imagining a generative design based on a simulation analogy, there are defined goals that spark the simulation. And even the parameters affected by the simulations are defined by the architect.

So, after all, answering the question 'can gen-AI fit in a professional architectural design process', of course. But it could be integrated into some phases of the process rather than starting the process. In the next section, some applications of gen-AI that could possibly add value to the process are suggested.

## 4.10 Generative and Non-Generative AI Usage Possibilities in Architectural Design

AI technology could be seen and thought of as a great tool for automating the design process which includes by nature visualization of ideas. Earlier in this chapter some of these applications were exhibited where a Gen-AI could generate a plan after defined boundaries (regardless of the idea that those boundaries were decided by AI in those examples).

Ali, S., 2020, argues that architectural visualization plays a crucial role in augmenting the comprehension of knowledge by minimizing cognitive load. The utilization of visualization tools enables individuals to grasp information more efficiently and to a deeper extent. By representing data in visual formats such as charts, graphs, or diagrams, complex concepts are simplified, aiding in quicker assimilation and enhanced retention. This visual approach leverages the brain's capacity to process and interpret images rapidly, allowing individuals to extract meaningful insights with greater ease. Whether conveying intricate datasets or illustrating abstract ideas, visualization serves as a powerful cognitive aid, facilitating a more intuitive and expedited understanding of information. Ultimately, the integration of visualization tools proves instrumental in optimizing the communication of knowledge across diverse fields.

However, and according to Ali, S., 2020, visualization in architecture has become a target more than a tool especially in architectural education. In fact, visualization could be misleading or deceptive. The beauty of a ‘hyper-realistically’ and aesthetically rendered glass box could mislead the client’s preferences. In this regard, visualization should be carefully dealt with by architects as a tool rather than as a product.

In terms of visualization, the idea of transforming sketches is seen very powerful and with more development it could make a great tool for visualizing plans, sections, other drawings, and perspectives when the machine has a 100% ability to generate images which apply exactly what is defined in a sketch. This application is seeing many developments today especially with the introduction of techniques like inpainting, outpainting, and Low-Rank Adaptation (LoRA). The more datasets to be fed to the generative algorithm, the more precise it will be in visualizing the architect’s sketch instead of generating new ideas.

Also, such technology could aid a lot in a phase of the design process called the ‘mood board’ in which architects search for inspirational designs and show it to the client in order to be on the same ground during the design phase. Such a phase is important before starting the concept design and is usually related to aspects like façade elements, aesthetical elements, and design style. However, this step is not meant to have a significant impact on the core of the form making/finding process.

Gen-AI image generation in this case, could have an added value based on the data it is trained on, and it is believed to have the same result as collecting the inspirational images from the web.

Additionally, Gen-AI applications regarding transforming images into 3D-models could have a huge impact on automating design tasks. Especially, if 3d-model gen-AI models developed to generate surfaces and clean meshes rather than point clouds or voxels. Such an application is not far from reality. It could develop through integrating the gen-AI model to a pipeline which exhibits an automated way to segment images and extract the main points' coordinates. Such an application could be exhaustive at first, but if better data sets are collected and engineered, it could not be regarded as impossible.

Finally, another bright application of Gen-AI is the video Gen-AI which could create animations and walkthroughs which are considered an architectural output in some projects. The idea of generating videos through photos collected from around a 3D-model using diffusion is now present and could be applied in such tasks.

On the other hand, and as discussed earlier in this chapter, non-gen-AI models do not generate data as images, text, 3d-models, etc. Instead, they are capable of predicting and generalizing on unseen data based on the pattern they learn during their training process. In this regard, non-gen-AI will not produce an image, but may predict numbers, or classes. These numbers and classes could be projected to the architectural field as parameters which could be used by the architects themselves or automated systems to generate products. This particular description aligns well with the approach taken to deal with the architectural process as a holistic system rather than a process of processes (described in chapter 1).

The idea of dealing with a building as a set of parameters which are interrelated and strongly connected in the design phase, could make good use of non-gen-AI applications. Accordingly, turning a building's design parameters to data sets which could be used to train AI and ML algorithms could yield many possibilities. Training a model with parameters either numerical or text to predict design decisions is thought to be an automation process saving time and effort for architects in the future. Imagine designing a cluster of buildings (a residential or administrative compound). Such projects could take months to create variations or prototypes of the building with different areas, functions, etc. but with the same architectural style and theme or either days but with more manhours or architects. With the aid of coding in extracting all the parameters and generating a data set including many designs with different areas that is used to train ML models for example, this could automate the 3d modeling tasks of different prototypes with different characteristics. Also, and looking from the same perspective, non-gen-AI models could be used to predict spatial relations and to detect proper spaces' areas based on learned data. Moreover, the models could predict and make decisions based on other aspects such as environmental and legislative aspects. For instance, they could predict a length parameter that defines the spacing between two staircases based on firefighting code or the tilt angle of louvers which reduce solar gain, or minimum required spacing that respects setbacks, etc. However, such decisions require neat and precise data sets so that predictions are mirroring real decisions based on real data. Those models are guaranteed to have learnt data directly from the architect.

In general, non-gen-AI models could be thought of the same way as generative-design techniques like optimization and simulations in the sense that the product is unknown, but it is still applying certain rules controlled by the architect through the data sets they learn from which could be very specific to the details-of-every-parameter extent and unique rather than general and repetitive. Thus, the overall process is controlled by the architect against any random decisions that could be made by gen-AI models.

**Summary**

This chapter provides a comprehensive exploration of the multifaceted definition of Artificial Intelligence (AI) and its transformative impact on technology. It begins by defining AI as the development of machines and systems capable of performing tasks traditionally requiring human intelligence, encompassing everything from rule-based systems to advanced neural networks.

Also, the chapter delves into the history of AI as well as its types and applications. The distinction between generative AI (Gen-AI) and non-generative AI (Non-Gen-AI) is explored, with profound implications in technical, ethical, and societal dimensions. Moreover, Machine Learning (ML), a subset of AI, is introduced as the backbone of intelligent systems, enabling machines to learn from data and improve performance over time. The symbiotic relationship between AI and ML is emphasized, with ML providing adaptive capabilities for AI to navigate dynamic environments. The chapter explores various ML approaches, including supervised learning with labeled data, unsupervised learning uncovering patterns in unlabeled data, and reinforcement learning where agents learn through trial and error. The versatility and applicability of ML methods across domains are highlighted. The chapter introduces a brief overview of specific ML algorithms used in the research and the evaluation metrics employed to assess trained ML models.

Moreover, different up-to-date applications of both Gen-AI and Non-Gen-AI in architecture and similar fields were presented. Gen AI algorithms showcase great innovation in the product. Most of the examples introduced using the machine as a designer, taking decisions, and forming a product. On the contrary, Gen-AI tools should be seen as assistants, providing architects with valuable insights and options rather than replacing the human element in design thinking.

In addition, the evolution of AI creative capabilities in the architectural field is explored from a skeptical perspective, particularly focusing on image generation models, which have progressed through various models and techniques, each introducing more options like inpainting, outpainting, and image-to-image generation. The impact of both generative and non-generative AI on the architectural design process is highlighted. Generative AI, known for its divergent thinking and algorithmic creativity, plays a pivotal role in idea generation and its reliability as a tool used in the architectural design process is questioned. Collaboration between architects and generative AI is emphasized as well as the architect's role in the process. In general, applying AI in the architectural field fits well with the grey box approach of design thinking. In a holistic design process composed of other processes that relate to each other cyclically, with today's design tools which benefit mainly from mathematics and physics, architects could think of some processes with a black box approach. In such processes, the embedded operations that happen form no concerns as the architect totally controls and directs them freely. In fact, the architect can direct and control Gen-AI systems, but it is concluded that the architectural design process cannot be diminished in a 2D-space with an image that hardly involves other design aspects. Such a process could lead to laziness and stripping the architectural design of its meaning. Also, the authenticity of AI-generated designs was discussed. From the understanding of how the models work, those models could be described as eclectic collage makers which present innovation through repeating elements that are learned from the training data set. Moreover, architectural visualization's direct influence is discussed as a useful tool using AI generative models.

Non-generative AI, on the other hand, excels in analytical support, aiding architects in data analysis, decision making, predicting, and classification tasks. The integration of AI in architecture, however, poses challenges, including ethical considerations, the balance between technological efficiency and human creativity, and the need for architects to adapt to evolving workflows to develop them rather than being a user. The chapter concludes by delving into benefits of using non-gen-AI models in the design process as automation assets to the architect in the decision making based on authentic and unique data provided by the architects themselves.

To conclude, this chapter reflects the dynamic and evolving nature of AI's role in architecture, ranging from opaque generative processes to transparent analytical tools. It is also concluded that many ML algorithms could aid in the architectural design process as well as architectural analysis studies and architectural education. In those applications, the AI process is human-centered, and AI is considered a tool for automatic heavy tasks using regression, classification, clustering, and RL algorithms.

# Chapter 5: Architectural Form Generation: Applying Machine Learning Algorithms on Architectural Parameters Datasets

**Preface**

In this chapter, a framework for utilizing ML in the form-generation process is explained. This process is considered form-finding in a sense that the models predict the parameters defining the form but is also considered form-making in a sense that it is made and decided by the machine according to how it learnt from the architect ed. Est., mapping the architect's decision-making black box approach. So, this process is better described in the space between form-making and form-finding being closer to form-finding literally.

This chapter defines the methods and tools used in the process. The coding process of the 3D-model to extract parameters is explained as well as how those parameters are related. After that, a resulting sample of the parameters data set is presented. And finally, how the full data set was generated is explained with the scope and limitations of the project.

Additionally, in this chapter exploratory data analysis is performed to two data sets to gain insights about the data. First, the parameters (features and targets) are correlated to each other and pre-process the data accordingly. After that, the data is resampled to either increase the number of samples or balance the data based on the problem. Then, the data sets are split to train and test sets before training different regression and classification models.

## 5.1 A Framework for Utilizing Machine Learning in Form Generation

This section presents a comprehensive framework that aims at automating the design process by predicting architectural design parameters using machine learning. By harnessing parametric modeling tools to create dynamic, data-rich design alternatives, and applying machine learning algorithms to analyze and predict optimal outcomes, this process transforms the way architectural decisions are made. The integration of computational techniques into the design process allows for more informed, data-driven choices, ensuring that design iterations are both innovative and feasible.

The following framework encompasses the key stages of creating a parametric building model, generating multiple design alternatives, and extracting critical parameters to form a dataset. From this data, machine learning models are trained, validated, and used to predict design parameters that meet specific goals. This framework not only enhances the efficiency of the design process but also paves the way for greater collaboration between architects and computational tools, facilitating a more integrated and intelligent approach to design.

1. Creating a Fully Parametric Model of the Building:

   In the early stages of the design process, the architectural model must remain flexible to allow for rapid iteration and adaptation to changing requirements. A parametric model is a dynamic representation of a building that can adjust its form and function in response to predefined variables, or parameters. These parameters typically include critical aspects of the building's geometry, such as height, width, depth, floor area, and window-to-wall ratio, along with structural considerations and façade treatments. Through parametric modeling, architects can explore a wide design space efficiently, ensuring that various design alternatives are responsive to both aesthetic and practical constraints. The parametric model is structured to allow for changes in variables, creating numerous design variations that can be analyzed later.

   The development of parametric models involves scripting techniques and parametric control setups in design software. Parametric modeling is grounded in algorithmic design, where relationships between design parameters are explicitly defined through code. Here, a system of inputs and rules governs the creation and modification of geometry, facilitating the automation of form generation.

2. Designing a Large Number of Well-Studied Alternatives

   Once the parametric model is established, it is used to explore a wide range of design alternatives. These alternatives are driven by different configurations of the model's parameters, reflecting variations in architectural expression, performance, and structural feasibility. The process ensures that each alternative aligns with certain project-specific criteria, such as the form’s tangible characteristics including proportions and areas, maximizing natural light, optimizing energy efficiency, or adhering to zoning regulations. This step is particularly critical in early-stage design, where exploring multiple design pathways helps stakeholders choose optimal solutions.

3. Extracting the Parameters of Each Alternative to Create a Large Dataset

   Each design alternative can be represented by a unique combination of parameters, such as floor area, building height, structural efficiency, and environmental performance. Capturing and recording these parameters is essential for creating a comprehensive design dataset. This dataset not only helps in understanding the design space but also allows for data-driven decision-making, making it easier to select or refine specific design outcomes based on quantitative analysis.

   The parametric design space can be encoded into a structured dataset, where each alternative is represented as a row and each parameter as a column. This data is stored in formats such as CSV, allowing for easy manipulation and analysis in various computational tools (Python, R, etc.). Each entry in the dataset reflects both independent variables (design parameters) and dependent variables (performance metrics), which will be critical for subsequent machine learning tasks. The structured format also allows for integration with external data sources, such as energy simulations, cost analysis, and environmental impact assessments.

4. Data Preprocessing

*Studying the Relation Between Parameters*

In architectural design, parameters are often interrelated. For instance, increasing the window-to-wall ratio might improve daylighting but could negatively affect energy performance due to heat loss. Understanding these relationships is crucial for informed design decisions. A thorough exploration of how various parameters influence each other, and the overall design is fundamental to ensuring that design alternatives are both functional and aesthetically pleasing.

This step involves the application of statistical techniques to understand the dependencies between parameters. Correlation analysis and visual tools such as scatter plots and heatmaps are employed to examine these relationships. In more complex scenarios, dimensionality reduction techniques such as Principal Component Analysis (PCA) can reveal underlying patterns within the data, helping to streamline the design space by focusing on the most influential parameters. These methods allow for a deeper understanding of how changes in one parameter influence others, offering insights that can inform both design and optimization strategies.

*Splitting the Dataset According to Parameter Relations*

Not all design alternatives will be equally relevant to the project's goals. By segmenting the dataset based on specific parameter groupings, designers can focus on the most critical areas of the design space. This step is vital for filtering out irrelevant or suboptimal alternatives, allowing the focus to shift to designs that meet certain thresholds for structural performance, sustainability, or user comfort.

Cluster analysis, such as K-means or Hierarchical Clustering, can be applied to partition the dataset into subsets based on parameter similarities. This clustering enables more efficient learning by training models on smaller, more homogeneous groups of data. Segmentation based on parameter relations ensures that each subset is more cohesive, allowing machine learning models to learn specific behaviors within subspaces of the design space.

*Balancing the Data through Resampling Techniques*

In architectural datasets, certain types of design alternatives may be overrepresented, leading to biased results when training predictive models. Balancing the dataset ensures that all design typologies and configurations are equally considered, leading to more generalizable and reliable predictions across the entire design space.

Resampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique) and undersampling are used to ensure that the dataset is balanced. This step is important in tasks where certain design outcomes might be rare, and thus underrepresented in the dataset. Balancing the dataset prevents the model from becoming biased towards more common outcomes, ensuring that minority designs are given equal consideration.

*Cleaning the Dataset*

Ensuring that the dataset is free from inconsistencies or errors is critical for meaningful analysis. Missing or erroneous values in the dataset may represent incomplete or faulty designs that should not be considered in the final analysis.

Data cleaning involves handling missing values, removing outliers, and standardizing the dataset. Missing data can be addressed through imputation (e.g., mean, median, or k-nearest neighbor imputation), while outlier detection methods can be employed to identify and remove extreme values that could distort the results. Standardization or normalization techniques may also be applied to scale the data appropriately, ensuring that all parameters are on comparable scales.

5. Training and Validating Machine Learning Models

*Training Machine Learning Algorithms*

Machine learning algorithms are trained to identify patterns in the data and predict new designs. These models learn from past design alternatives, allowing architects to predict the behavior of new designs based on historical data. This process helps architects to quickly evaluate a wide range of alternatives, reducing the need for manual exploration.

Various machine learning algorithms are applied, depending on the problem at hand. For regression tasks (e.g., predicting continuous variables like energy consumption or structural stability), algorithms such as Linear Regression, Random Forests, and Gradient Boosting are used. For classification tasks (e.g., categorizing building types or design styles), algorithms like Support Vector Machines (SVM) and Neural Networks are employed. The models are trained using cross-validation techniques to avoid overfitting and to ensure that they generalize well to new data.

*Choosing a Champion Model Based on Validation Metrics*

The selection of a "champion" model is based on how well it predicts desired design outcomes, balancing accuracy with interpretability. The chosen model should not only provide accurate predictions but also align with the designer's intuition and architectural goals.

Validation metrics such as R-squared, Mean Squared Error (MSE), Accuracy, and F1-Score are used to evaluate the performance of the trained models. Cross-validation is used to compare different models and select the best one (the "champion") based on its performance across multiple datasets. This model will be used to make predictions in the next phase of the design process.

6. Predicting, Parsing Parameters & Fine-Tuning

*Using the Champion Model to Predict the Parameters*

Once trained, the model can predict new design parameters based on user-defined criteria. For example, an architect may specify that a building should maximize daylight while minimizing energy consumption, and the model will predict the optimal set of parameters that satisfy these goals. Additionally, an architect may specify the area of the building, and the model will predict the lengths and widths of the floor slabs.

The chosen model is deployed to predict the design parameters for new building scenarios. Input features may include high-level goals, such as cost limits or sustainability targets, or tangible goals like the built-up area of the building, walls offset from slabs, etc. and the model will output a set of parameter values that reflect the best design solution based on learned patterns in the data.

*Parsing the Predicted Parameters to the 3D Model in Modelling Software and Finetuning them*

The predicted parameters are reintroduced into the parametric model, allowing for the real-time generation of 3D geometry that reflects the machine learning model's suggestions. This process creates a seamless connection between data-driven predictions and the physical form of the design, enabling architects to immediately visualize how the suggested parameters translate into architectural space.

By updating the 3D model in real-time, architects can rapidly iterate on the design, assessing both its functional and aesthetic qualities. The parametric nature of the model allows for flexibility—if certain parameters need adjustment for practical or creative reasons, they can be fine-tuned directly in the design software. This direct feedback loop helps maintain the balance between algorithmic optimization and architectural intuition, ensuring that the machine-generated designs remain responsive to the unique contextual and human factors inherent in architectural projects.

For example, if a machine learning model predicts optimal window-to-wall ratios based on energy efficiency goals, these parameters can be directly applied to the 3D model in the 3D modelling software. The facade's geometry will automatically update, allowing the architect to visualize and evaluate the design from multiple perspectives, considering light distribution, visual impact, and user comfort, among other factors. This integration facilitates a more interactive and informed design process where data enhances, rather than replaces, the designer's expertise.

The suggested framework integrates parametric modeling and machine learning to optimize architectural design. A fully parametric model generates multiple design alternatives, which are analyzed by extracting key parameters into a dataset. After preprocessing the data—studying parameter relationships, splitting the dataset, balancing, and cleaning—machine learning models are trained to predict optimal design parameters. The best-performing model is selected, and its predictions are parsed back into the 3D parametric model. Finally, the design is fine-tuned, combining data-driven insights with architectural judgment to achieve the desired outcomes.

Although the architect’s decisions (way of thinking) could be ill-defined or ill-structured, this framework should give results which stick to the pattern found within the architect’s decisions expressed in the choice of parameters which affect the outcome.

Figure 5-1 explains the suggested framework to utilize ML in the form-finding process.

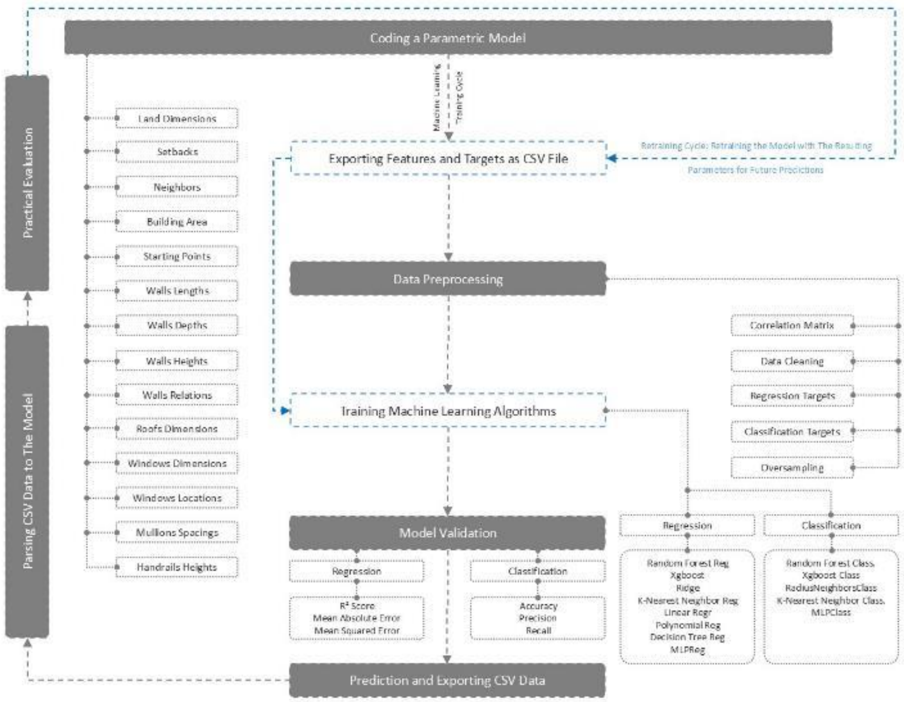



*Figure 5-1- Suggested Machine Learning framework for Regression and Classification Tasks in Architectural Modeling - By the Author*

Following, the suggested framework is applied to a contemporary villa design to validate it.

## 5.2 Problem Definition, Scope, and Limitations

The problem in this project is described as analyzing whether ML models could map the architect's way of thinking and -metaphorically- take decisions based on patterns that might exist in his black-box described thoughts or not.

To test this case, a villa is designed parametrically in a contemporary style for the sake of simplicity. Modeling this villa by coding resulted in full control over the parameters to generate 600 samples (consciously designed) and ease of transforming them into a ML-ready data set.

The project is applied to give the user (architect) full control of requirements including a rectangular land's length and width, total built-up area, setbacks, four neighbor types, and number of building blocks.

Those parameters directly decided the form-generation process by affecting tangible parameters including each slab's length and width, recess and dimensions between slabs, windows' existence and widths, shading devices numbers and width, and walls relations to each other. Only tangible parameters were tested for simplicity.

The generated data set was used to train different ML models to test whether they find the pattern that was consciously presented by the designer or not. Those models included a variety of regression and classification models which are compared to tests which would suit this type of problems. So, the target was not to develop an ML model but to test them and test the reliability of the generated data set.

To evaluate the results (predictions) of the used ML models, regression results are evaluated based on $R^2$ score, MSE, MAE, and RMSE. And to evaluate classification results, accuracy, precision, recall, and f1 scores were used as evaluation metrics. Also, a classification report is generated for each classification model. Multiple regression and classification metrics scores are used to assess the results.

## 5.3 Methods and Tools

Modeling the parametric villa is done using C# component on Grasshopper v. 1.0 for Rhinoceros3d v.7 implementing the RhinoCommon's API.

The analysis in this study is performed using the Jupyter Notebook v.6, and Python v.3.10. Scikit-learn library v.1.3.0 for Python was used for ML and TensorFlow v.2.10.1 and Keras v2.10 were used to train ANNs.

Also, NumPy v.1.23.5 was used to manipulate data as arrays during the process and Pandas v.2.0.3 was used to manipulate data in a CSV[1] format as data structures. Additionally, for data visualization Matplotlib v.3.7.2 and Seaborn v.0.12.2 libraries for Python were used.

[1] CSV stands for "Comma-Separated Values." It is a simple and widely used file format for storing tabular data, such as a spreadsheet or database, in a plain text format.

PC Specifications: 16 GB RAM, Intel (R) Core (TM) i7-8700K CPU @3.70GHz, Nvidia GTX 1060 3GB.

The tools used in this study were selected for their efficiency and suitability in handling complex architectural modeling and machine learning tasks. C# in Grasshopper with RhinoCommon API allows for precise control of parametric models, essential for creating 3D models. Jupyter Notebook and Python offer an interactive environment for analysis, while Scikit-learn provides robust machine learning algorithms for parameter evaluation. TensorFlow and Keras are ideal for training artificial neural networks, crucial in design optimization. NumPy and Pandas streamline data manipulation, and Matplotlib with Seaborn enhances data visualization, ensuring clear insights throughout the process.

## 5.4 Coding an Architectural Design Model

In this section the modeling of the villa by coding using C# is explained and how the parameters were related to affect each other and to build a consistent model is discussed. Finally, a resulting sample of the data set is presented.

### *5.4.1 Modeling the project*

A villa prototype is modelled in C# language using RhinoCommon API (figure 5-2). To model the villa, walls and slabs were created as boxes with a starting point (x, y) that varies depending on the design and is directly affected by the setbacks. Walls are interconnected to each other with ruling parameters that define where a wall starts and how long is a wall offset from another one. Windows are modeled by cutting in the walls using the boolean command and mullions are added as boxes spaced by dividing the window width to an integral number parameter. The parametric model allows the user to parametrically change the land width, land length, neighbors (street/neighbor), setbacks, number of blocks for the villa, starting point, walls widths, walls lengths, walls heights, slabs lengths, slabs widths, floors heights, whether there is a window in each wall or not, width of window in each wall, and window's center point distance from the wall's center point. The number of building blocks is either 2 or 3 to facilitate modeling villas with larger areas.

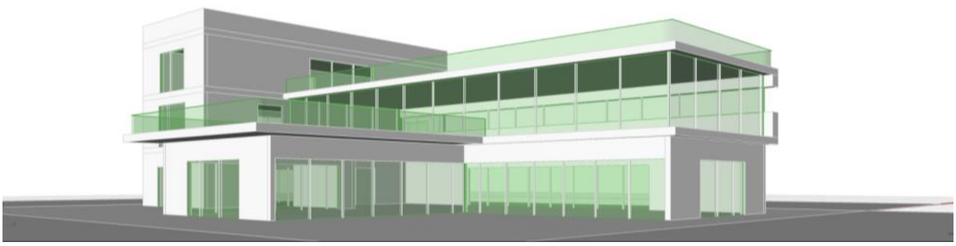


*Figure 5-2 – Villa prototype coded in C# on Grasshopper for Rhinoceros3d*

The slabs widths and lengths are ruled mathematically to not exceed the setbacks.

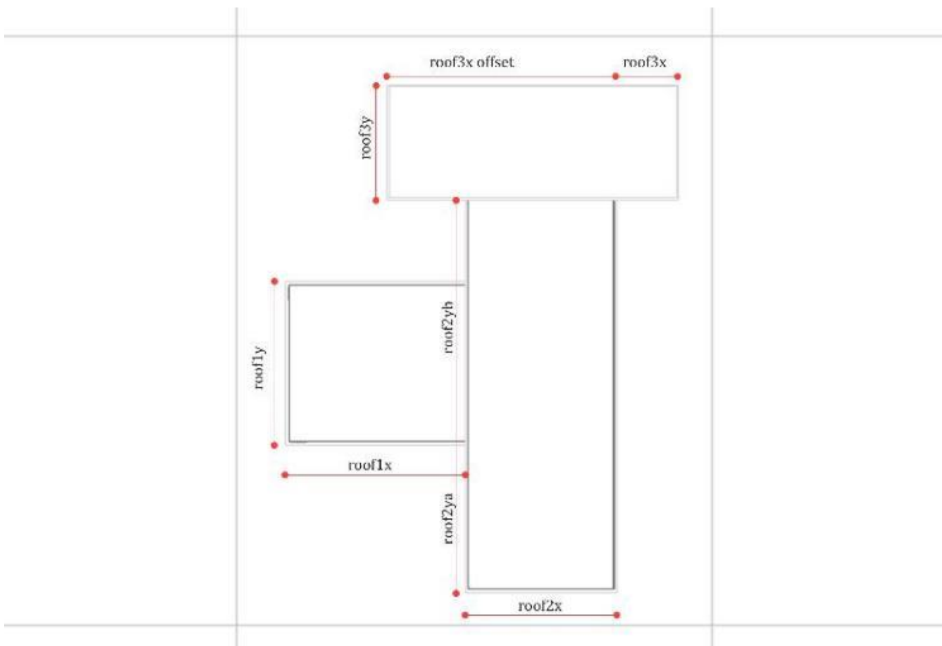



*Figure 5-3 – Slabs Annotations*

And changing the slabs dimensions affects the area of the villa which is calculated by the equation:

$$Area = (roof1x \times roof1y) + \big((roof2x \times (roof2yb - roof2ya)\,) \times 2\big) + \big(((roof3x\,offset - roof3x) \times roof3y\big) \times 3)$$

Or

$$Area = (roof1x \times roof1y) + \big((roof2x \times (roof2yb + |roof2ya|)\,) \times 2\big) + \big(((roof3x\,offset + |roof3x|) \times roof3y\big) \times 3)$$

Variables in the equation are shown in figure 5-3.

### *5.4.2 Parametric relationships*

The starting point 'SP' is the base of the coded model. This starting point is affected by the setbacks and is conditioned to have x and y values that are equal to or greater than the setbacks values. At the same time, it starts shaping roof 1, which is modeled with intervals starting from the starting point and heading towards the x and y directions. Roof 2 is parametrically connected to roof one. Its starting point is roof 1's point 'rf1b' and its y dimension is 'roof2ya' + 'roof2yb' which are mathematically related to roof 1's point 'rf1b'. In the case of having three building blocks, roof 3's starting point is roof 2's point 'rf2b'. The x dimension of roof 3 is equal to 'roof3x' + 'roof3x offset'. The point between the two lines 'rf3b' has the same x coordinate as point 'roof2$_{b.}$' Block 1 is modeled to have only one floor, block 2 is modeled to have two floors, while block 3 is modeled to have three floors. Figure 5-4 shows points annotations.

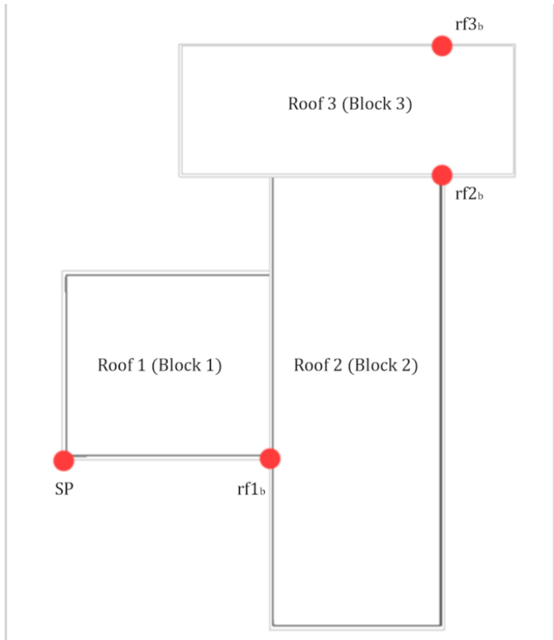



*Figure 5-4 – Points annotations of the villa*

The walls are modeled in relation to the floors and to each other. So, a wall can be offset from the roof's edge or falls exactly on its edge. This requires a relationship between each two perpendicular walls where a wall's length could vary depending on the other walls' exact location.

Also, the windows were parametrically built in each respective wall with parameters regarding the center of the window relative to the center of the wall and how windows' lengths should not exceed the respective wall's length. And conditions were added to choose whether there is a window in a wall or not with true/false values.

Handrails are also modelled with mathematical relations with the roofs edges so that they can be offset from the roofs or on the roofs' edges. Their shapes also change automatically depending on the relation between the building blocks.

### 5.4.3 Resulting sample

Using coding, a CSV file is automatically created to receive the parameters of the villa. And automatically again by coding, the parameters are transferred to the file by pressing a button in grasshopper canvas. The sample shape has two rows and 125 columns, and the set looks as shown in table 5-1:

*Table 5-1 - Sample Data Shape*

| **Land Area** | **Land Length** | **Land Width** | **Setbacks(X)** | **Setbacks(Y)** | **Built-Up Area** | **Number of Blocks** | **Neighbor 1** | **Neighbor 2** | **Neighbor 3** |
|---|---|---|---|---|---|---|---|---|---|
| 408 | 17 | 24 | 3 | 3 | 215 | 2 | Street | Neighbor | Neighbor |
| **Neighbor 4** | **Starting Point(X)** | **Starting Point(Y)** | **GF Height** | **GF Wall 1 X-Offset** | **GF Wall 1 Y-Offset** | **GF Wall 2 Y-Offset** | **GF Wall 4 X-Offset** | **…** | **SF Wall 12 Y-End-Offset** |
| Neighbor | 3 | 10 | 3 | 0.4 | 0.7 | -5 | 1.8 | … | 0 |
| **GF Roof 1 X** | **GF Roof 1 Y** | **GF Roof 2 X** | **GF Roof 2 YA** | **GF Roof 2YB** | **GF Roof 3 X** | **GF Roof 3 X-Offset** | **GF Roof 3 Y** | **…** | **SF Lintel Height** |
| 5 | 7 | 7 | -15 | 8 | 0 | 13.9 | 9 | … | 2.2 |
| **GF Wall 1 Window** | **GF Wall 1 Window Width** | **GF Wall 1 Window Offset** | **GF Wall 2 Window** | **GF Wall 2 Window Width** | **GF Wall 2 Window Offset** | **GF Wall 3 Window** | **GF Wall 3 Window Width** | **…** | **SF Wall 15 Window Offset** |
| TRUE | 4 | 0 | TRUE | 5.1 | TRUE | 6.4 | TRUE | … | 0 |

## 5.5 Generating A Machine-Learning Ready Dataset

In this section, the generated data sets are analyzed and explained in detail.

The 2 created data sets are composed of 600 samples of villas designs. Each data set has different targets to be predicted. The targets of the 2 data sets are the parameters used to create the villa's model using C# coding in grasshopper3d. Figure 5-5 shows the resulting model of the target parameters.

*Figure 5-5 – Resulting villa model which was built by using parameters in code.*

Each 100 samples are designed to have a specific land area, and setbacks options. Several villas' areas are designed within a specified range and the samples are divided into 4 categories where neighbor types are changed which affected the parameters. Table 5-2 shows samples numbers and parameters that affected the designs.

*Table 5-2 - Numbers of samples and parameters that affect the designs.*

| Total Number of Set | Number of Elements | Land Area | Villa Area | Number of Blocks | Land Length | Land Width | Samples Number / Neighbor Types | | | | Setbacks |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | South Street and 3 Neighbors | West Street and 3 Neighbors | South and West Streets and 2 Neighbors | South, West and East Streets and 1 Neighbor | |
| 600 | 100 | 400-500 | 200-250 | 2 | 20 | 23 | 25 | 25 | 25 | 25 | 2-3 |
| | 100 | 520 | 250-350 | 2 | 20 | 26 | 25 | 25 | 25 | 25 | 2-3 |
| | 100 | 644 | 350 - 450 | 2/3 | 23 | 28 | 25 | 25 | 25 | 25 | 2-3 |
| | 100 | 836 | 450 - 550 | 2/3 | 22 | 38 | 25 | 25 | 25 | 25 | 2-3 |
| | 100 | 910 | 550 - 650 | 2/3 | 26 | 35 | 25 | 25 | 25 | 25 | 2-3 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 100 | 1120 | >650 | 3 | 28 | 40 | 25 | 25 | 25 | 25 | 2-3 |

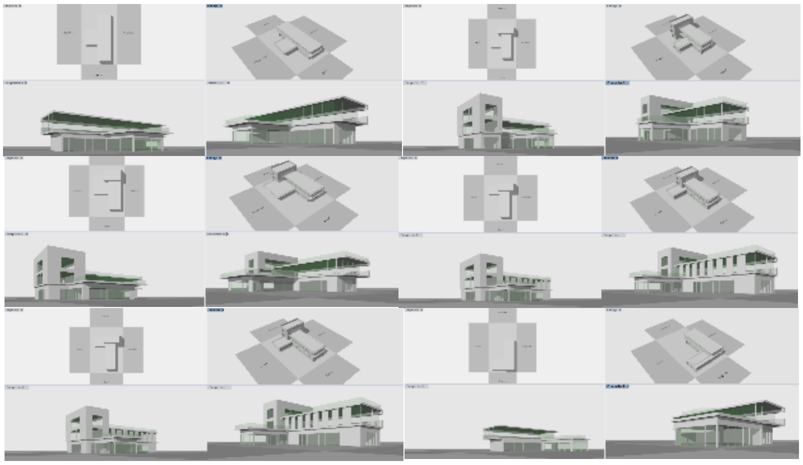


*Figure 5-6 - Screenshots of random samples of the data set*

### *5.5.1 Data set 1 (form data set):*

In the first part of this data set (areas data), the features including land dimensions, total built-up area, number of blocks, and setbacks are used to predict the slabs locations as well as their dimensions, and the building's starting point (targets). The slabs widths and lengths are ruled mathematically to not exceed the setbacks.

Changing the slabs dimensions affects the area of the villa which is calculated by the equation:

$$Area = (roof1x \times roof1y) + \left((roof2x \times (roof2yb - roof2ya)) \times 2\right) + \left(\left((roof3x\,offset - roof3x) \times roof3y\right) \times 3\right)$$

The second part of this data set (rest of parameters data) involves parameters related to distances between walls, walls' locations measured from slabs edges, and number of shading devices added to the first floor. Those variables are predicted according to the same features from the first part in addition to neighbor types.

### *5.5.2 Data set 2 (windows data set):*

In the first part of this data set, the features are the same as used in data set 1 while the targets are whether there is a window or not in a wall. The second part of this data set has the same features as the first part, but the targets are the widths of the windows in walls.

## 5.6 Data Correlations

Heat maps are created to thoroughly explore the correlations between the variables in the regression data sets (figures 5-7 and 5-8) and through a point-biserial correlation the correlations between the variables in the classification data set is explored (figure 5-9).

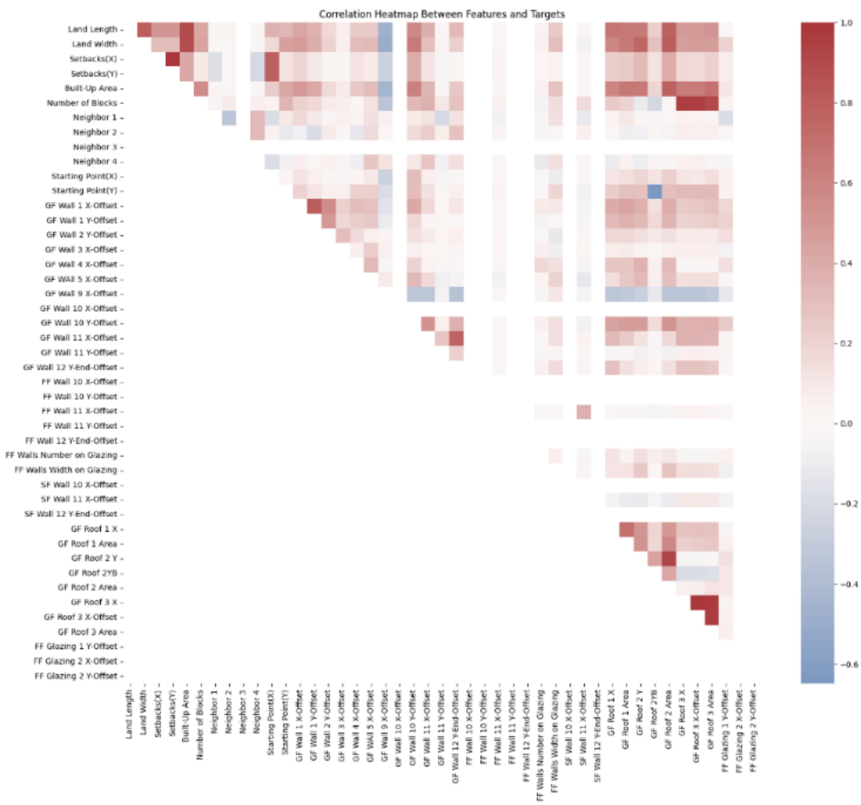



*Figure 5-7 – Correlation heatmap between features and targets for the data set 1*

From this correlation matrix, some variables that were consistent along the data set could be noticed. These variables should be removed. Also, drawing from the matrix, it could be also found that some variables could cause misguidance to the ML including some variables with negative values. Also, many features are not directly affecting many targets and vice versa.

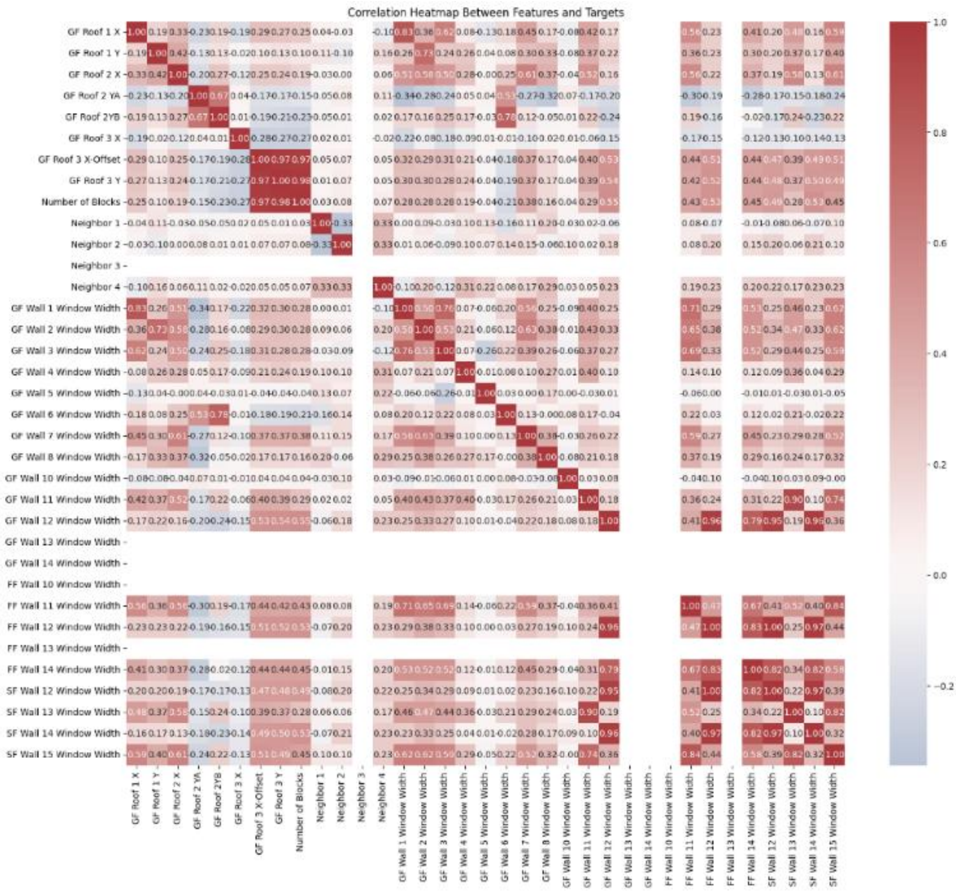



*Figure 5-8- – Correlation heatmap between features and targets for dataset 2 (regression)*

The same is noted in this data set with some variables being consistent along all the samples. Also, high correlation is noticed between some windows width and the aligning slab length.

Also, when plotting a point-biserial correlation between features and binary targets in data set 2, some consistent variables that need to be dropped are found.

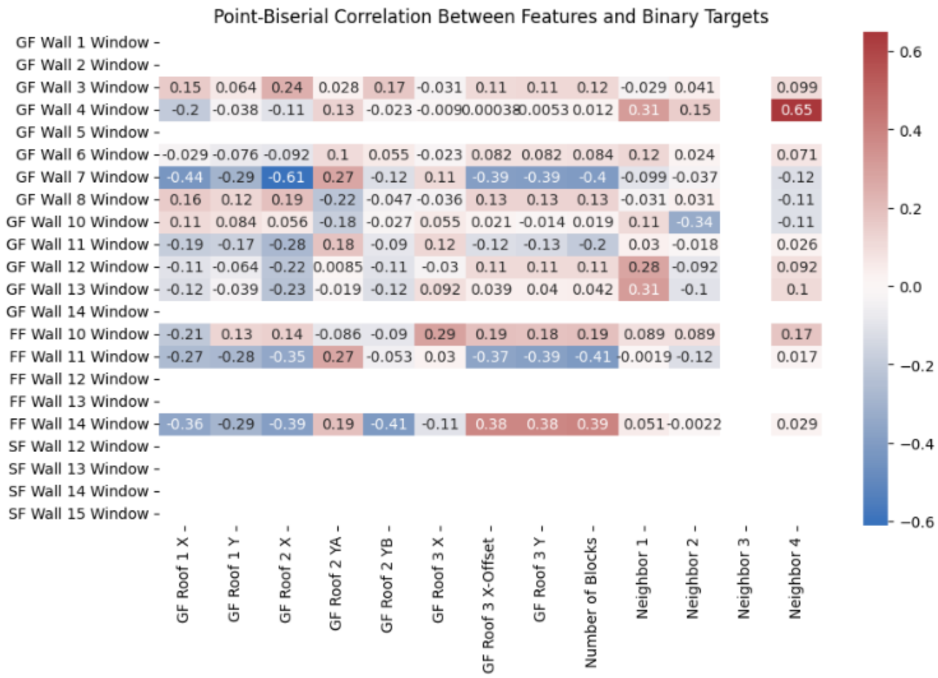



*Figure 5-9 – Point-Biserial correlation between features and binary targets in data set 2 (classification)*

## 5.7 Data Pre-Processing

The two original data sets under investigation consist of 94 columns containing all the parameters mentioned in section 9.4.

Data set 1 has 45 columns while data set 2 has 57 columns. However, both data sets share 10 columns which are supposed to serve as features including the land dimensions, built-up area, neighbor types, and setbacks.

Each data set has 600 samples designed by changing the parameters and screenshot of the villa design depending on the parameters set is saved in the same order as the csv data set for later practical validation (figure 3). Those 600 samples are designed carefully, and the parameters were changed intentionally depending on the architect's satisfying results regarding proportions, neighbors, areas, etc. This may ensure that a particular pattern exists within the data which the machine could observe.

### *5.7.1 Cleaning continuous data*

To achieve high efficiency in areas results after prediction, some of the highly correlated variables needed to be adjusted. Instead of predicting roof1x, roof1y, roof2x, roof2ya, roof2yb, roof3x, roof3x offset, and roof3 y, and to ensure logical training, some of these variables are replaced with more relevant data that can ensure the areas prediction accuracy and can mathematically output the replaced data. For example, roof1y was replaced with roof1 area, roof 2 ya which happens to have a negative value usually was replaced with roof2 area, and roof3 y was replaced with roof3 area. This process was important to make the numeric pattern clearer to the algorithms. Figure 5-10 shows roofs labels. However, this process was not done on the windows data set because the lengths and widths of walls played a key role in defining the windows' width because walls lengths were mostly related to slabs dimensions.

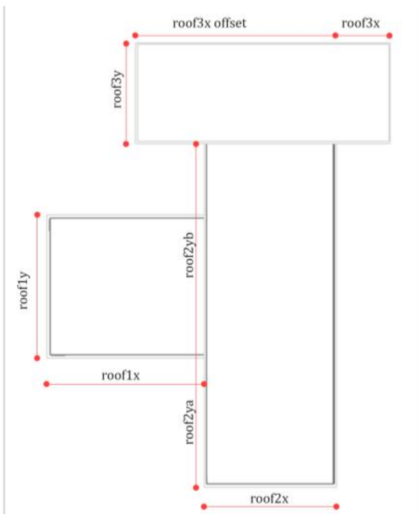



*Figure 5-10 - Roofs labels*

In the form data set, some targets are found consistent among the data set and are dropped to make it easier for the machine to find relationships. From the features, neighbor 3 was dropped from both data sets for always having a 'neighbor' value set as 0. Also, setbacks in the X dimension and in the Y dimension were found to be similar to each other in all of the samples so, setbacks Y was dropped.

### *5.7.2 Cleaning categorical data*

Some of the data in the set are of a string type like neighbor and street. And some are boolean with 'True' and 'False' values like whether a window exists in a wall or not. Text in these cases is converted to numerical values for ML algorithms compatibility and achieving efficient computation. In this case, both 'true' and 'street' values are set to '1' while 'false' and 'neighbor' values are set to '0'. Moreover, some targets are dropped for having consistent values in all samples.

The results of data cleaning led to the areas and rest of parameters data set to have 33 columns, and the windows data set to have 42 columns.

## 5.8 Data splitting and choosing features and targets

Features in ML are the input data that the user gives to predict some values. The values to be predicted are called labels or target variables. In this problem, and as mentioned in section 9.4, the features and targets are chosen as shown in table 5-3:

*Table 5-3 - Features and Targets of The Model*

| Features (Inputs) | | Targets (Outputs) | |
|---|---|---|---|
| Feature | No. of Features | Target | No. of Targets |
| Land length | 1 | Starting point | 2 |
| Land width | 1 | Walls parameters | 22 |
| Setbacks | 1 | Slabs dimensions | 8 |
| Built-up area | 1 | Windows existence | 22 |
| Number of blocks | 1 | Windows widths | 22 |
| Neighbors | 4 | Shading Devices Number | 1 |
| | | Shading Devices Widths | 1 |

Initially, the features include what an architect should input to the program to output the numerical parameters that control the architectural aspects like proportions, dimensions of walls, and windows, etc.

The inputs include the building area, land width, land length, number of blocks, neighbors, and setbacks. The dataset is split to a form dataset where form parameters are predicted such as walls offsets, slabs dimensions, starting point of the form, etc. and another “windows” dataset for predicting windows existence as well as windows dimensions.

### *5.8.1 Form data set splitting*

The correlation matrix between features and each column in targets is plotted. The correlation matrix was plotted as a heatmap using Seaborn library in Python and some variables were excluded from the data frame for a cleaner dataset.

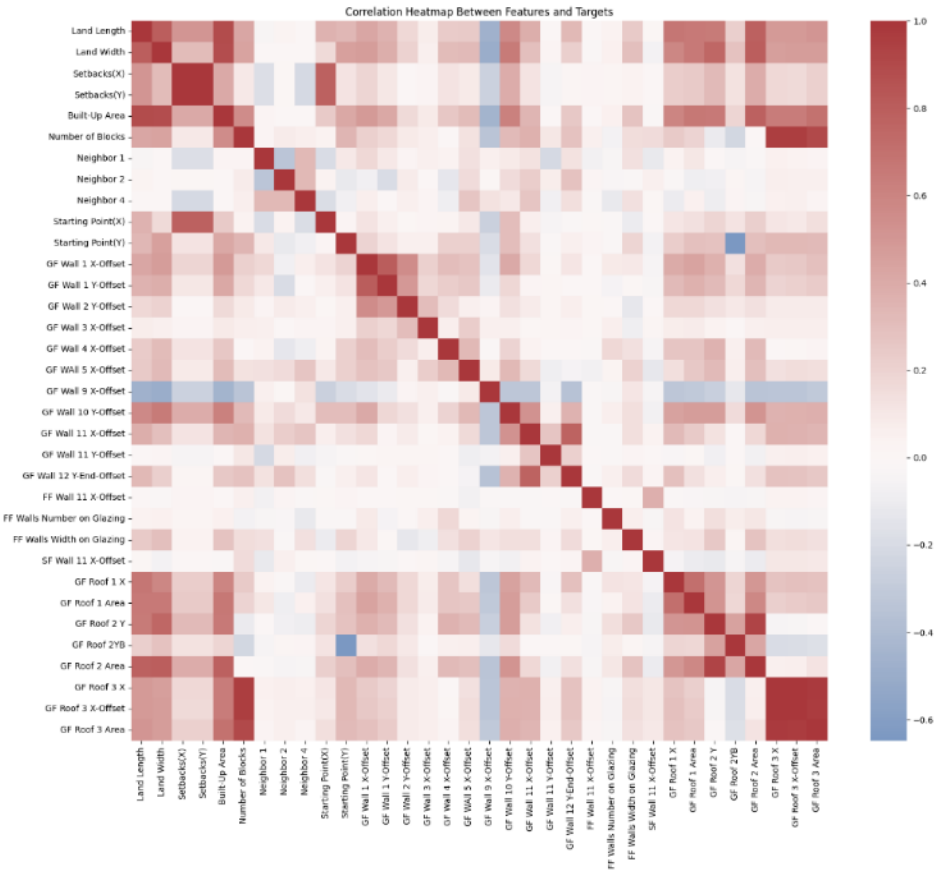



*Figure 5-11 - Heatmap showing correlations between the variables in the dataset (By the Author)*

The plot shows that only a few features which could affect the building area represented in slabs dimensions. Those include land length, land width, built-up area, and number of blocks.

For this reason, the form dataset is split to areas related dataset to predict the slabs dimensions thus, the building area, and another dataset to predict the walls offsets, and other form related parameters.

The features of the first data set are land length, land width, setbacks, built-up area, and number of blocks.

While the other data set includes all the targets and features of the cleaned dataset to train the model for predicting the rest of form parameters using features like land length, land width, setbacks(X), built-up area, number of blocks, and the types of the neighbors. Figures 5-12 and 5-13 show the correlations between each dataset's variables.

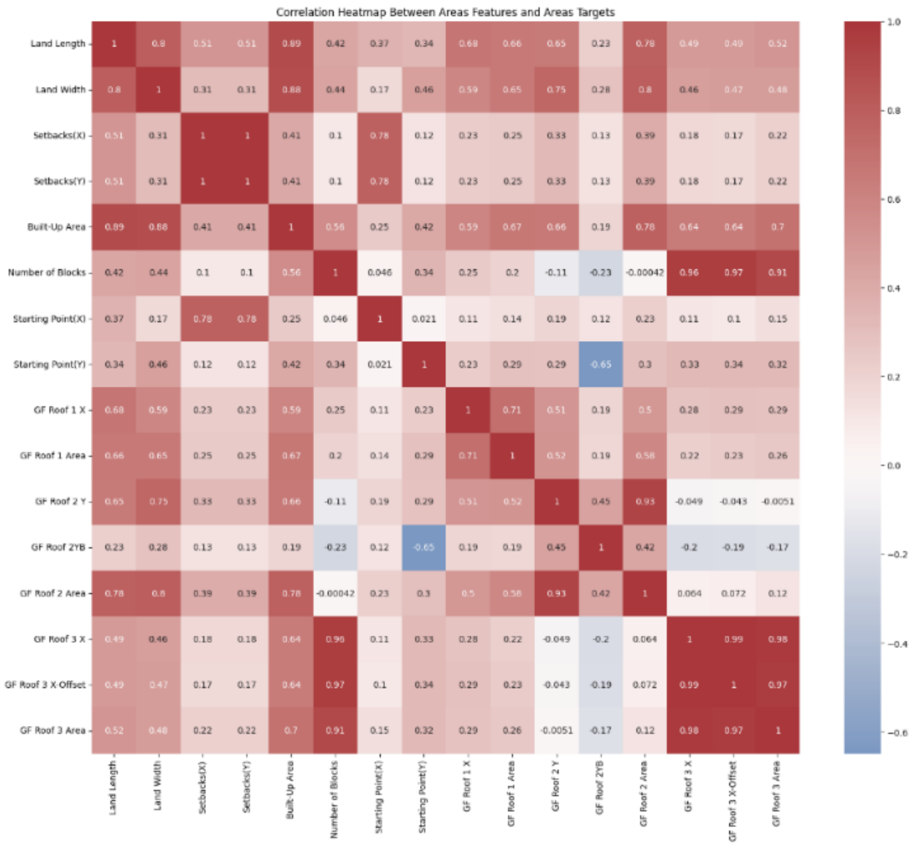



*Figure 5-12 - Heatmap showing correlations between the variables of the form areas data set (By the Author)*

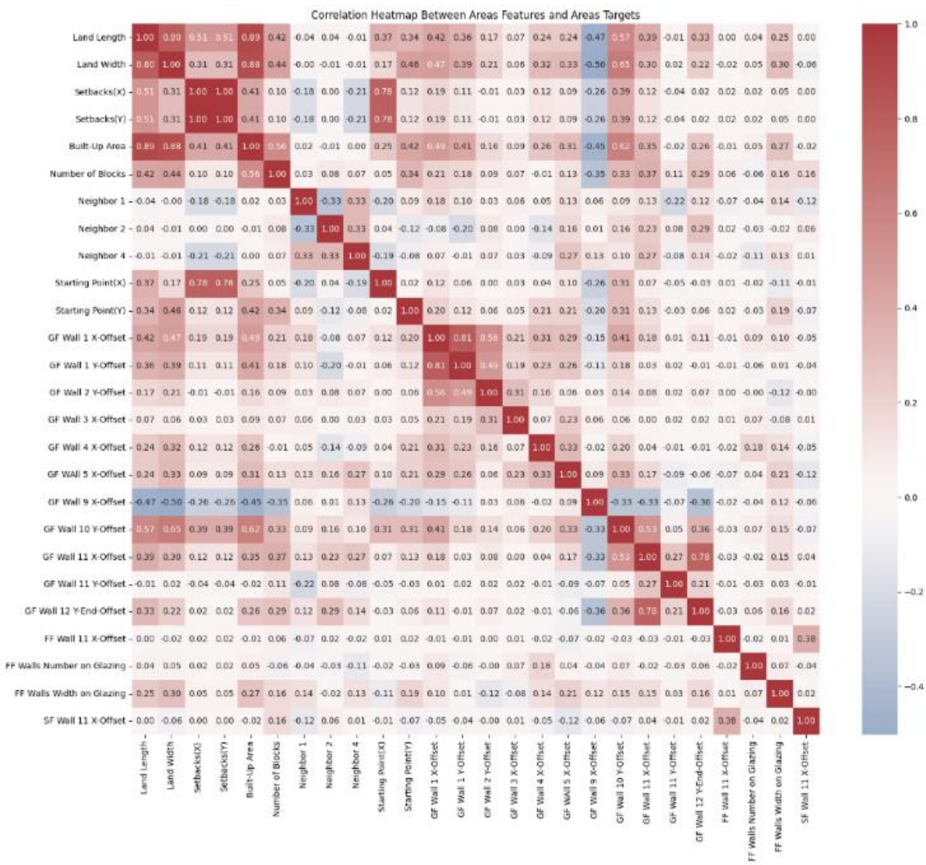



*Figure 5-13- Heatmap showing correlations between the variables of the form rest-of-parameters data set (By the Author)*

### *5.8.2 Windows data set splitting*

The windows dataset includes 12 features which are the neighbor types, the number of blocks, and the slabs' dimensions which logically points to whether a window exists in a wall or not and the width of the window (in the designs and dataset, the walls were directly correlated to the slabs dimensions). The targets are the windows' width and whether a window exists in a wall or not. This shows two types of targets, one is boolean and one is continuous. Those two types require classification and regression models respectively. So, the targets were split to classification targets and continuous targets. Figure 5-14 shows the correlation matrix heatmap between continuous targets and features while figure 5-15 shows Point-Biserial correlation between binary targets and features.

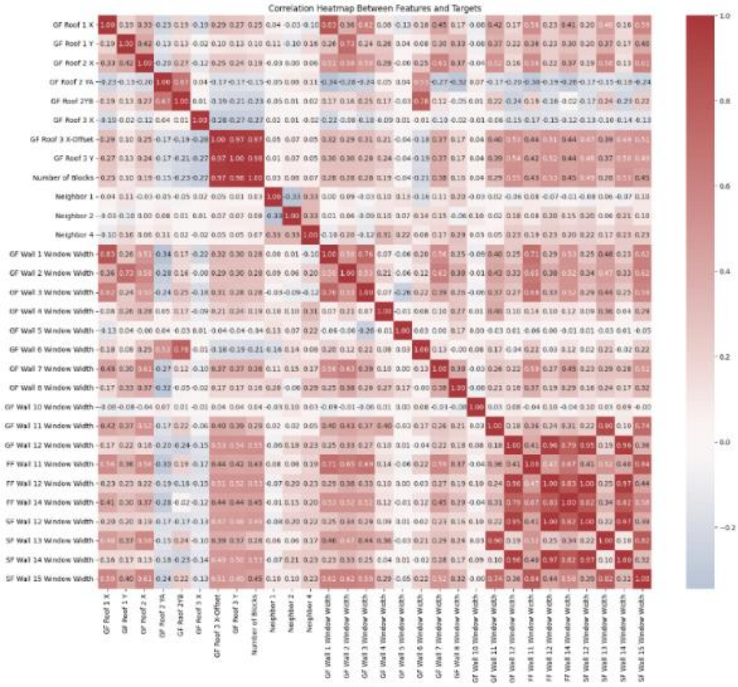


*Figure 5-14 - correlation matrix heatmap between continuous targets and features*

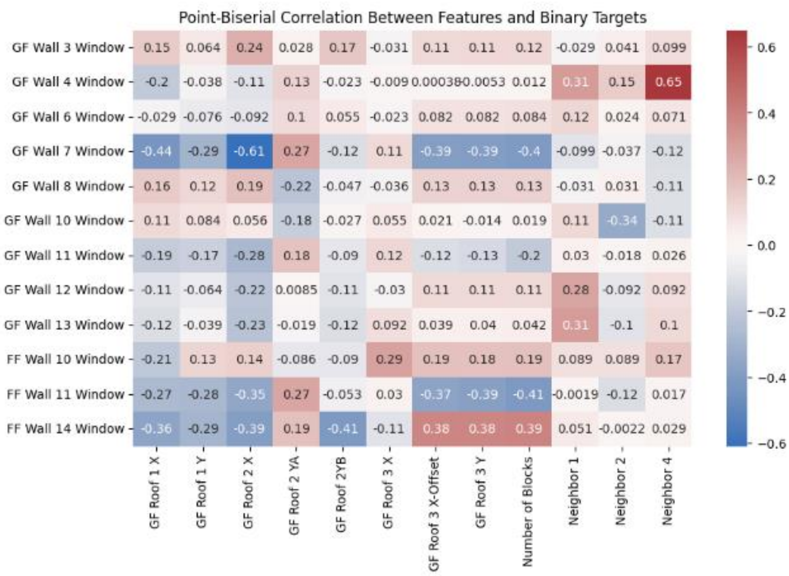


*Figure 5-15 - Point-Biserial correlation between binary targets and features*

Both the correlation heatmap and point-biserial correlation map give us precise thoughts of how the dataset was generated. Regarding the binary targets, clear correlations exist between the neighbors and the slabs' side and the window in the wall which lies under the slabs side and facing this exact neighbor. Also, the correlation matrix heatmap reads well in the same manner.

### 5.9 Data Resampling

Resampling in ML refers to the process of creating a new dataset by either duplicating instances (oversampling) or removing instances (undersampling) in order to achieve a more balanced class distribution. The goal is to improve the performance of ML models, particularly in cases where one class is significantly underrepresented compared to another.

There are only 600 samples of designs that were created by the architect which could be insufficient for ML to analyze the data. To produce enough samples for ML training, employ general-purpose resampling functionality in Sklearn library which performs oversampling in this case.

150 of the 600 samples have neighbor 1 value as '0', and 450 have neighbor 1 value as '1'. Also, 150 samples have neighbor 2 value as '0', and 450 have neighbor 2 value as '1'. So, neighbor 1 and neighbor 2 as neighbors ('0' value) are considered minority classes while the two columns as streets are considered dominant classes. In this case the resample technique was used to oversample the minor classes to balance the class distribution. The resulting data set includes 30,600 samples for the areas' data set and 30,600 for the windows' regression data set.

*Synthetic Minority Oversampling Technique (SMOTE)*

Regarding the windows classification data set, a significant imbalance in data was noticed, which was a result of the architect's preferences. Figure 5-16 shows count plots of windows classification targets before applying SMOTE.

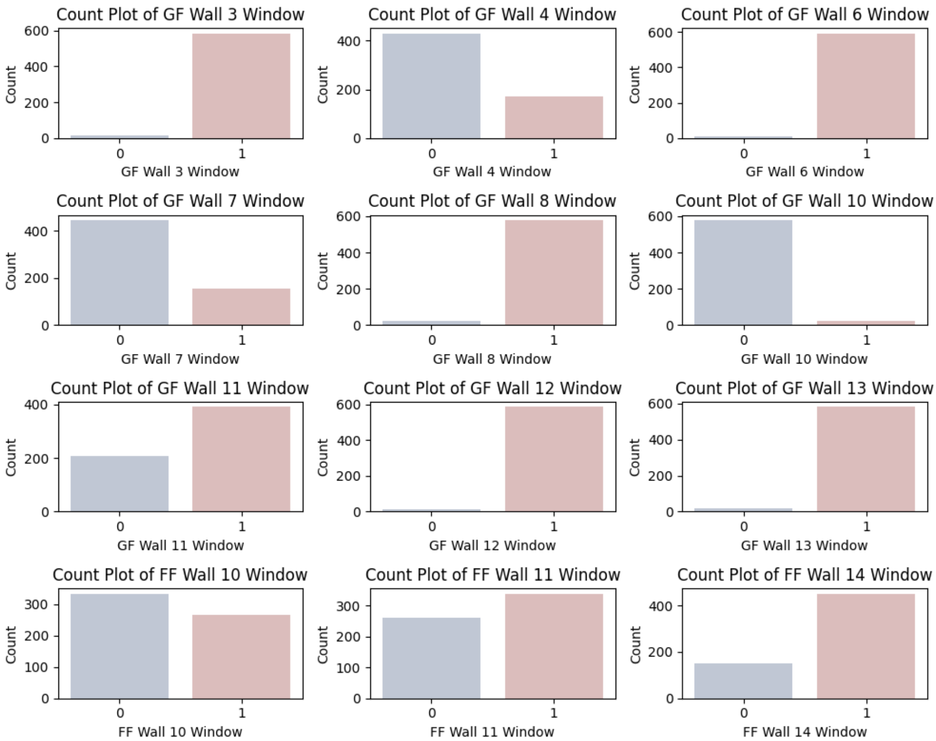



*Figure 5-16 - Count plots of targets of the windows data set before applying SMOTE*

The same resampling technique used with the areas' data set was not able to balance the data. SMOTE was applied to this data set. It is a technique that generates synthetic samples for minority classes to tackle imbalanced data sets. SMOTE primarily operates within the feature space, creating new instances by interpolating between closely positioned positive instances.

Figure 5-17 shows count plots of windows classification targets after applying SMOTE.

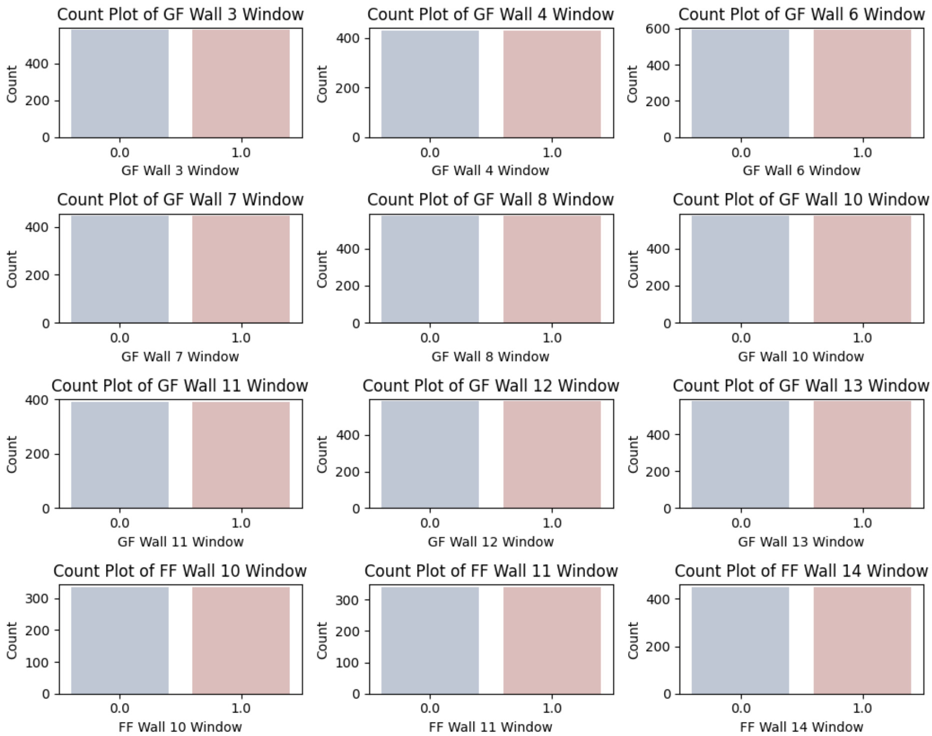



*Figure 5-17 - Count plots of targets of the windows data set after applying SMOTE*

## 5.10 Train-Test Splitting

To perform a ML task, data should be split to at least a training set and a test set. The test size in this case is set to be 20% of the whole dataset while the remaining 80% are assigned for the training set. The data is shuffled before splitting to get a test set with randomized instances representing the entire data set. K-fold cross validation was applied.

## 5.11 Training Models

After having preprocessed and cleaned data, ML techniques were applied to predict architectural design parameters of the villa model. The problem involves both regression and multi-class classification.

There are three regression tasks, one for predicting the areas and slabs dimensions, one for predicting the architectural parameters including walls offsets, and one for predicting the windows' widths and one classification task which predicts the existence of windows in different walls. Table 5-4 shows different problem types in this study.

*Table 5-4 – Machine learning problem types in this study*

| **Prediction Tasks** | **Problem Type** | |
|---|---|---|
| | Regression | Classification |
| | Slabs dimensions and starting point | Existence of a window in each wall |
| | Architectural parameters | |
| | Windows' width | |

Different regression and classification models were applied directly using scikit-Learn, TensorFlow and Keras modules for Python. All these models were discussed earlier in section 4.3.: Random Forest, XGBoost, Ridge, K-Nearest Neighbor, Linear Regression, Polynomial Regression, Decision Tree, and Multi-Layer Feedforward Neural Network (MLP) in TensorFlow. All the ML regressors were used with their default parameters while the MLP regressors hyperparameters' were tuned to get the best possible scores.

For the MLP, the best trained ANN consisted of 3 layers: an input layer of 64 perceptrons and a 'Relu' activation level, a hidden layer with 64 perceptrons and a 'Relu' activation level. And an output layer with 15 perceptrons and a 'linear' activation level. The loss was calculated based on the mean squared error and the used optimizer was 'Adam'. The batch size used in training was 32 and the number of epochs was 100.

A set of classification algorithms are used to predict windows existence in walls including Random Forest, k-NN, SVC, Decision Tree, AdaBoost, and XGBoost. All the ML classifiers were used with their default parameters.

Table 5-5 shows trained models used in this project.

*Table 5-5- Different models trained for regression and classification tasks*

| **Trained Models** | |
|---|---|
| **Regression** | **Classification** |
| Random Forest | Random Forest |
| XGBoost | XGBoost |
| Ridge | AdaBoost |
| k-NN | k-NN |
| Linear Regression | Support Vector Classification (SVC) |
| Decision Tree | Decision Tree |
| Polynomial Regression | |
| MLP-Scikit-Learn | |
| MLP TensorFlow | |

## Summary

In this chapter an application as a framework is introduced to utilize ML in the form-finding/making process, the problem was explained, being exploring a way to test ML algorithms on architectural design parameters as data sets to see how well ML models could generalize and find patterns that are consciously made by the author. Then the materials and methods used to perform this study were explained starting from choosing software, programming languages, and programming modules to analyze the data and build the models. It is also discussed in detail how the 3d-model was built using coding by C# language and how all the parameters were related together to make a fully parametric contemporary villa where every parameter could change other parameters with full control over them to automate the generation of a data set which contains only numerical and text values as variables. This data was split into two data sets. One to explore the parameters related to the form-generation and the other to control the parameters deciding windows' creation.

Additionally, the chapter delved into analyzing the data set by drawing correlations between variables. Such correlations are essential to understand how variables correlate to each other. The result of this phase led to splitting the large data set into two data sets, each having variables that directly affect each other, making it easier for the machine to learn form and conclude patterns. Further splitting is done to each data set according to the same concept where some variables were crucial in determining the area of the building while others affected parameters related to walls placements and formation. Also, in the second data set, some parameters fit into a classification task while others required a regression task. So, the data set ends up split into four sub-data sets. The sub data sets were cleaned from variables which were consistent along the whole data set and some variables were mapped to numerical values. Furthermore, the analysis of data balance was crucial in the study. Some techniques including SMOTE for classification and Oversampling for regression data sets were used to balance the data. After that, the four data sets were split into train and test data sets with an 80-20% ratio respectively. Finally, different regression and classification models were trained on the four data sets to gain insights on which models were more suited to this specific problem.

In the next chapter, the results of the training phase are explained and analyzed.

# Chapter 6: Machine Learning Analysis and Results

**Preface**

In this chapter the results of the different ML models are presented, analyzed, and compared. Additionally, the model learning of both the form data set and the windows data set based is discussed and evaluated on the evaluation metrics discussed earlier. Also, how predictions are done after training the models and choosing the best model are discussed. After that, a fine-tuning phase is proposed to give the architect control over the ML outcome to modify the built 3d-model easily to become a natural part of the design process. Finally, final conclusions are drawn from the results of the project.

## 6.1 Feature importance

According to the findings, the building area - as suggested - was the most important feature in the final areas' dataset (as shown in Fig. 6-1).

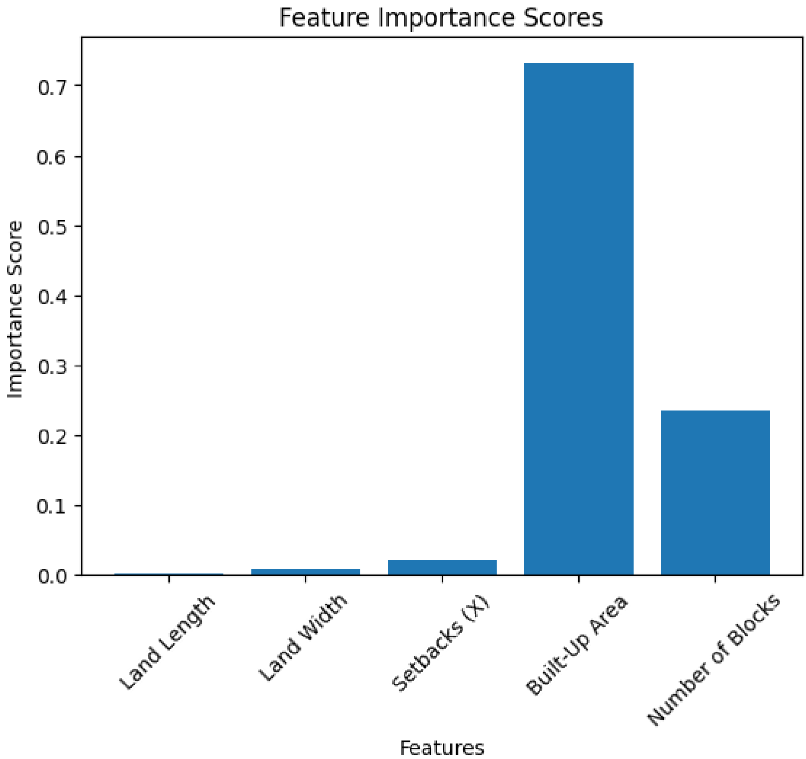



*Figure 6-1 - Feature Importance Score for a Random Forest Regressor*

This feature was directly affecting the areas of the floor slabs which is calculated based on the lengths and widths of the slabs. While the land length, land width, and setbacks were affecting only two targets which are 'Starting Point X' and 'Starting Point Y' representing the x and y coordinates of the starting point.

## 6.2 Evaluation Metrics

There are some metrics that show how an ML algorithm performs while training and with unseen data as well. These metrics vary according to the problem type, either a regression model or a classification model.

### 6.2.1 Evaluating regression performance

A regression model's performance is evaluated using metrics such as $R^2$ score, mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE).

$R^2$ score measures the proportion of variation in the dependent variable (Y) that is explained by the independent variable (X). $R^2$ score is calculated by finding the sum of residuals squared and the total sum of squares. The sum squared regression is the sum of the residuals squared, and the total sum of squares is the sum of the distance the data is away from the mean all squared. This is a critical measure for assessing model fit, with values ranging between 0 and 1. An $R^2$ score $\geq 0.9$ is considered excellent, $> 0.8$ is good, and $\geq 0.6$ can be acceptable in some scenarios, although there might be noticeable predictive errors. An $R^2$ score $\leq 0.5$ indicates poor explanation of data variation and potential limitations in prediction. MSE evaluates how well the regression model fits the data and its square root provides an estimate of the standard deviation ($\sigma$) of the random error term. Although RMSE is not an unbiased estimator of $\sigma$, it remains a dependable tool for this purpose. These metrics primarily measure the magnitude of regression errors but do not provide insights into the explained portion of the variance. MAE is the average absolute error between actual and predicted values. (Equations are shown in appendix B).

### 6.2.2 Evaluating classification performance

Accuracy, precision, recall, and F1 scores are evaluation metrics for classification tasks. Also, a classification report is typically generated. These scores can assess the performance of a classification model.

Accuracy represents the ratio of accurate predictions to total guesses. Precision relates to a classifier's capability to correctly classify a negative instance as negative. Sensitivity/recall, often called the true positive rate, gauges the model's proficiency in identifying all positive occurrences in relation to the combined count of true positives and false negatives. F1 score is the harmonic means of precision and recall. The harmonic mean is a mathematical average derived by dividing the total number of observations or elements in a series by the reciprocal of each individual number within that series.

The harmonic mean of accuracy is the F1 score. The number of actual instances of the class in the provided dataset is referred to as recall and support. (Equations are shown in appendix B).

## 6.3 Model learning analysis

ML analysis was conducted on the two data sets.

### *6.3.1 Form data set analysis*

To estimate the slabs dimensions and the rest of parameters, 8 ML models were conducted which are: random forest, XGBoost, Ridge, k-NN, linear regression, polynomial regression, decision tree, and MLP with TensorFlow. A total of 30,600 samples were used (24,480 samples for training and 6,120 samples for testing) for each model.

According to the findings, ensemble learning models had the best results among other algorithms. The results were very accurate especially when done on the areas sub-data set and the predicted slabs dimensions parameters succeeded to give very close values of total-built up area as required in the input. Figure 6-2 and table 6-1 show a comparison between the regression metrics between the 9 trained models.

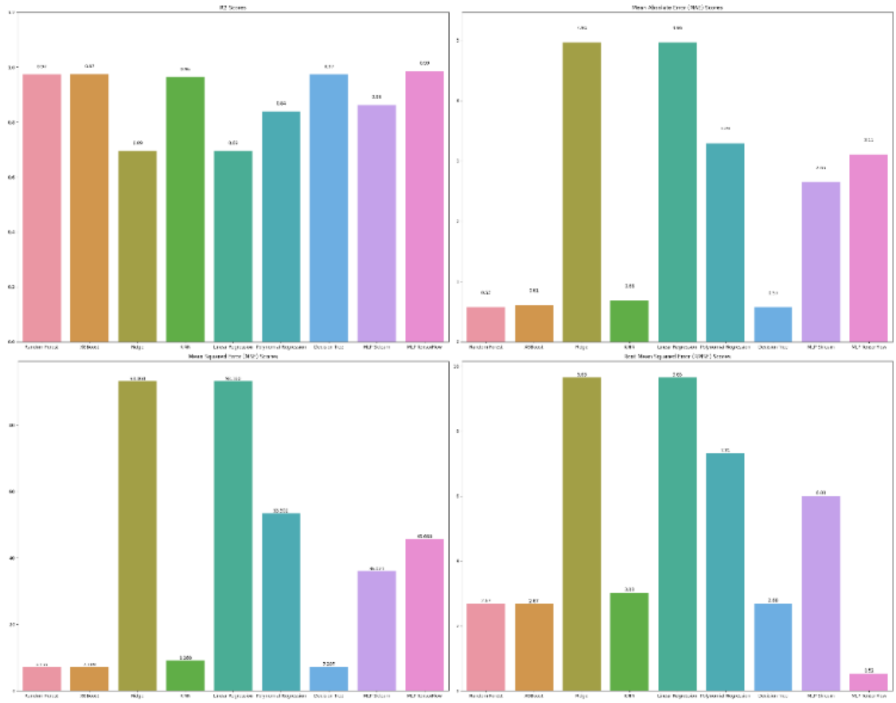


*Figure 6-2 - Comparison of the regression metrics between the 9 trained models for the areas sub-data set.*

*Table 6-1- Comparison of the regression metrics between the 9 trained models for the areas sub-data set*

| Model | R2 Score | MAE | MSE | RMSE |
|---|---|---|---|---|
| Random Forest | 0.97 | 0.57 | 7.15 | 2.67 |
| XGBoost | 0.97 | 0.61 | 7.11 | 2.67 |
| Ridge | 0.69 | 4.96 | 93.16 | 9.65 |
| k-NN | 0.96 | 0.68 | 9.17 | 3.03 |
| Linear Regression | 0.69 | 4.96 | 93.16 | 9.65 |
| Polynomial Regression | 0.84 | 3.29 | 53.50 | 7.31 |
| Decision Tree | 0.97 | 0.57 | 7.21 | 2.68 |
| MLP-Scikit-Learn | 0.86 | 2.65 | 36.02 | 6.00 |
| MLP TensorFlow | 0.99 | 3.11 | 45.67 | 0.52 |

In addition, the graph in figure 6-3 shows the training loss and validation loss during training the Feed-forward MLP. The graph shows convergence to almost 0 in both training and validation during epochs.

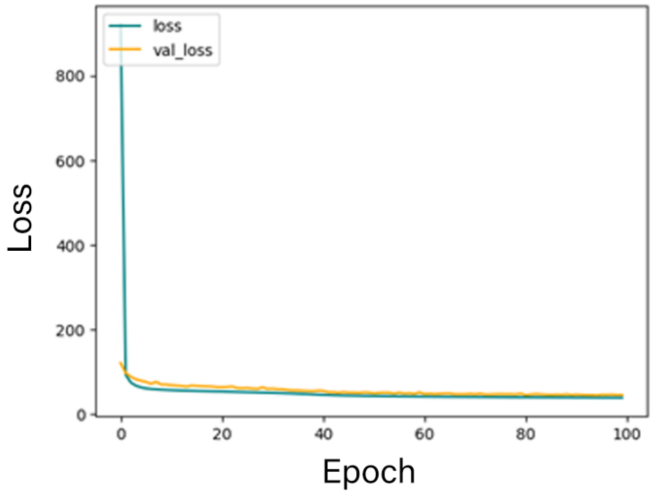



*Figure 6-3- Training loss and validation loss during training the Feed-forward MLP on the areas regression sub-data set*

Also, figure 6-4 shows a scatter plot with the best-fit regression line. This graph shows neither signs of overfitting nor underfitting.

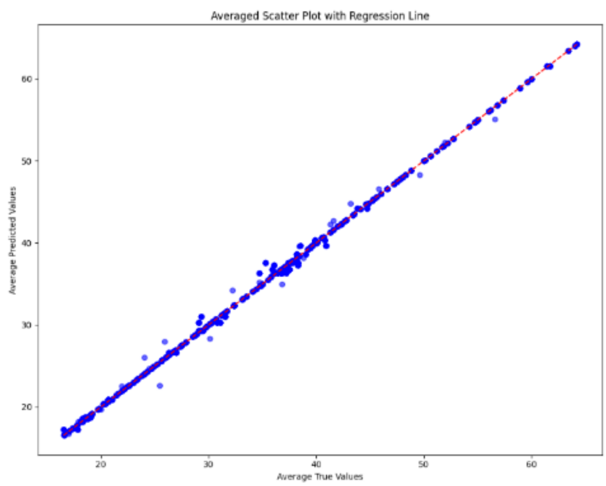



*Figure 6-4 - Averaged scatter plot on test data set with the best-fit line Created by Random Forest Regressor (areas sub-data set)*

Even with the rest of parameters sub-data set, ensemble models yielded much better results than other algorithms like ridge, linear regression, and polynomial regression. However, k-NN algorithm result was close to ensemble learning algorithms. Figure 6-5 and table 6-2 show a comparison of the regression metrics between the 9 trained models for the rest of parameters sub-data set.

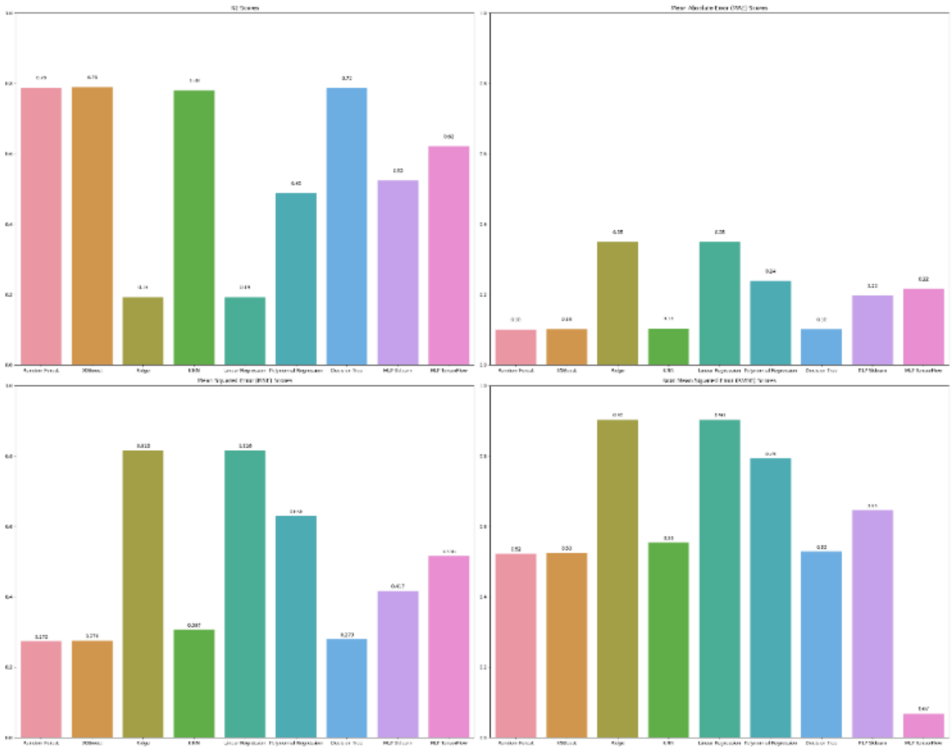


*Figure 6-5 - Comparison of the regression metrics between the 9 trained models for the rest of parameters sub-data set.*

*Table 6-2- Comparison of the regression metrics between the 9 trained models for the areas sub-data set*

| Model | R2 Score | MAE | MSE | RMSE |
|---|---|---|---|---|
| Random Forest | 0.79 | 0.10 | 0.27 | 0.52 |
| XGBoost | 0.79 | 0.10 | 0.28 | 0.53 |
| Ridge | 0.19 | 0.35 | 0.82 | 0.90 |
| k-NN | 0.78 | 0.10 | 0.31 | 0.55 |

| | | | | |
|---|---|---|---|---|
| Linear Regression | 0.19 | 0.35 | 0.82 | 0.90 |
| Polynomial Regression | 0.49 | 0.24 | 0.63 | 0.79 |
| Decision Tree | 0.79 | 0.10 | 0.28 | 0.53 |
| MLP-Scikit-Learn | 0.52 | 0.20 | 0.42 | 0.65 |
| MLP TensorFlow | 0.62 | 0.22 | 0.52 | 0.07 |

The graph in figure 6-6 shows the training loss and validation loss during training the Feed-forward MLP. The graph shows convergence to almost 0.20 in training while convergence in validation loss was higher reaching almost 0.53.

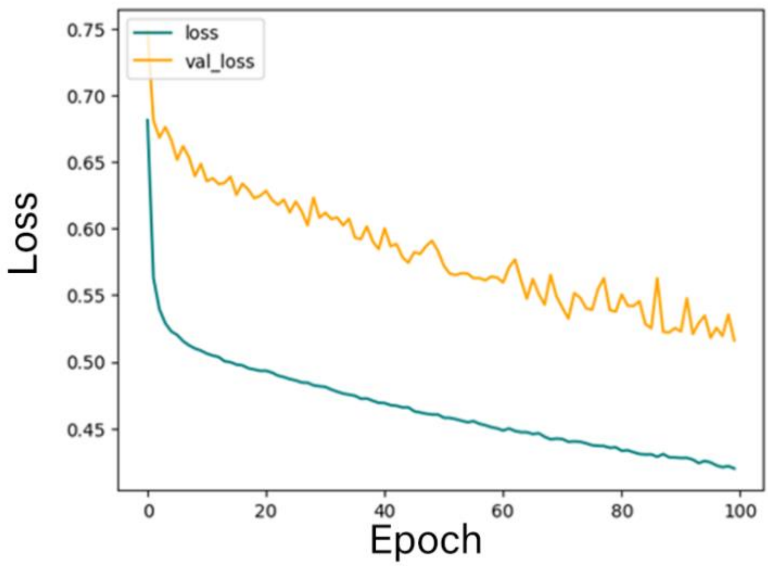



*Figure 6-6- Training loss and validation loss during training the Feed-forward MLP on the rest of parameters regression sub-data set*

Overall, the random forest algorithm performed exceptionally and had the best metrics results.

Figure 6-7 shows the best-fit line, visualizing the relationship between the average true values and average predicted values from the random forest regressor for all targets in the rest of parameters sub-data set which also show neither overfitting nor underfitting although some outliers appear but generally the line looks well-generalizing the relation.

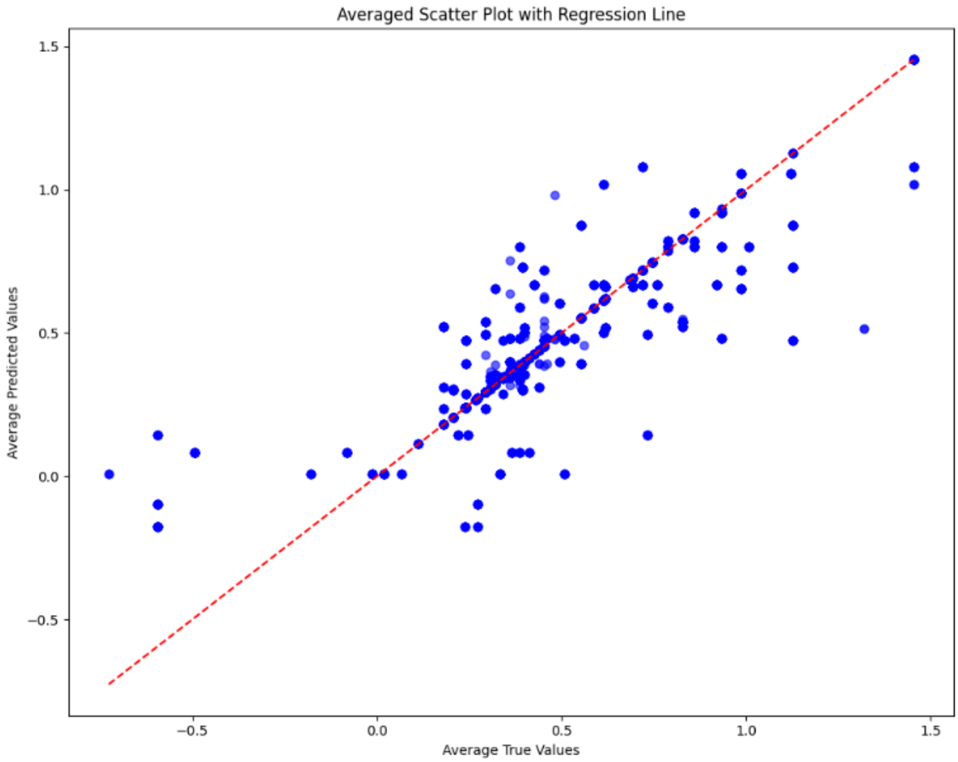



*Figure 6-7 - Averaged Scatter Plot with Best-Fit Line Created by Random Forest Regressor the rest of parameters sub-data set*

The results of the first data set are highly promising in terms of evaluation metrics.

### *6.3.2 Windows data set analysis*

The windows data set featured two problems which are a classification problem to detect whether a window exists in a wall or not and a regression problem to predict the windows' widths.

Various ML algorithms were trained on both sub-data sets. In regression, 7 algorithms were trained which are: random forest, XGBoost, ridge, k-NN, linear regression, decision tree, and MLP. All the ensemble learning algorithms (random forest, XGBoost, and decision tree) performed exceptionally well and their scores were very close. Even k-NN performed well and very close to the mentioned algorithms. Only linear regression and ridge models underperformed with a low $R^2$ score of 0.49 for each. Figure 6-8 and table 6-3 show comparison of scores between the 7 algorithms used with the windows width sub-data set.

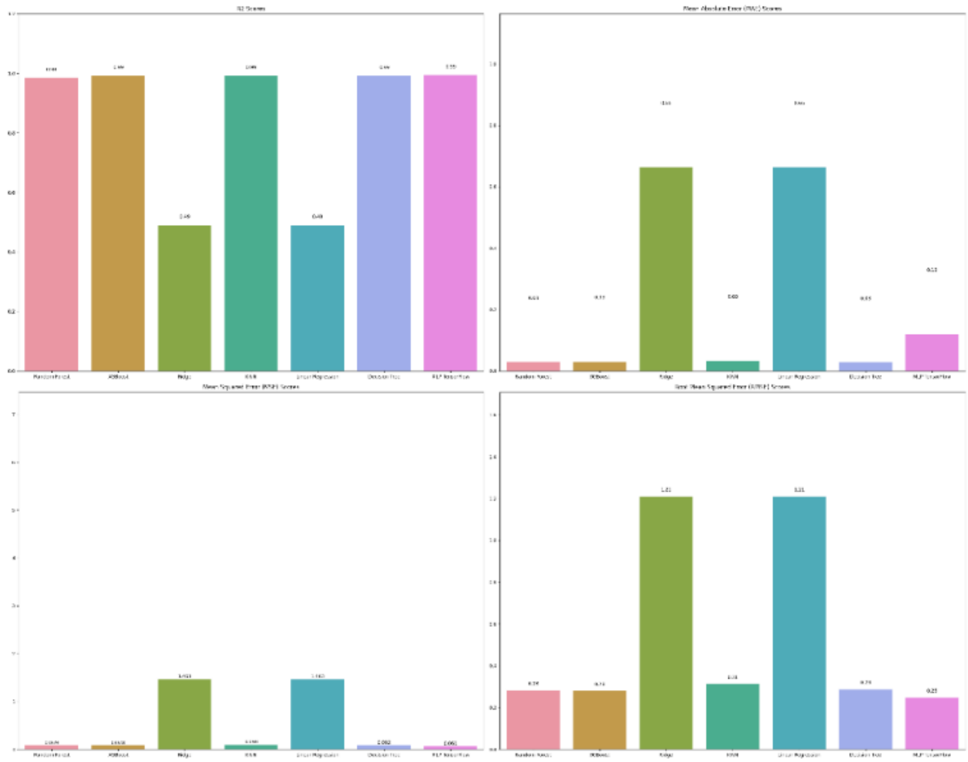


*Figure 6-8- Comparison of scores between the 7 algorithms used with the windows width sub-data set*

*Table 6-3- Comparison of the regression metrics between the 7 trained models for the windows width sub-data set*

| Model | R2 Score | MAE | MSE | RMSE |
|---|---|---|---|---|
| Random Forest | 0.98 | 0.03 | 0.08 | 0.28 |
| XGBoost | 0.99 | 0.03 | 0.08 | 0.28 |
| Ridge | 0.49 | 0.66 | 1.46 | 1.21 |
| k-NN | 0.99 | 0.03 | 0.10 | 0.31 |
| Linear Regression | 0.49 | 0.66 | 1.50 | 1.21 |
| Decision Tree | 0.99 | 0.03 | 0.08 | 0.29 |
| MLP TensorFlow | 0.99 | 0.12 | 0.06 | 0.25 |

Moreover, the feed forward NN also performed very well with a loss value that started with 0.92 and dropped to as low as 0.06. However, its MAE score was higher than k-NN and ensemble learning algorithms.

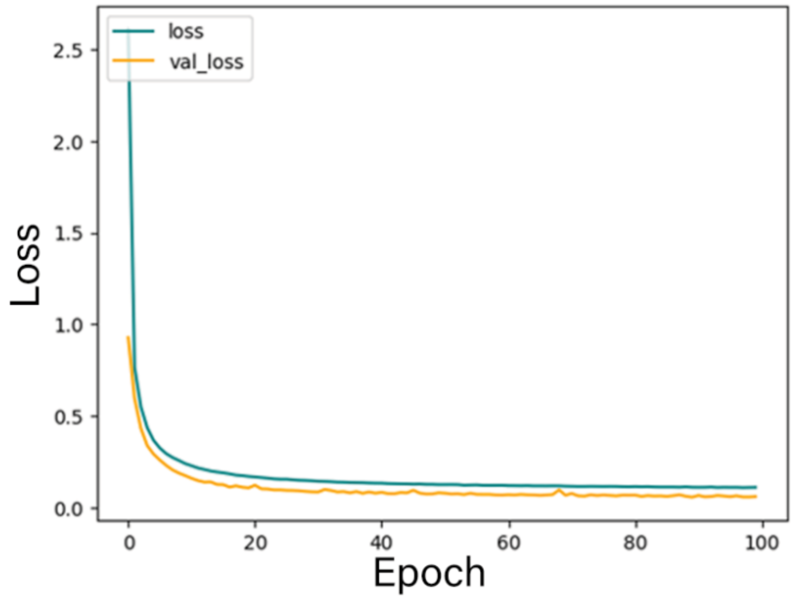



*Figure 6-9- Training loss and validation loss during training the Feed-forward MLP on the windows width regression data set*

Overall XGBoost achieved the best results. Figure 6-10 shows the best-fit line, visualizing the relationship between the average true values and average predicted values from the XGBoost regressor for regression targets in the windows widths sub-data set.

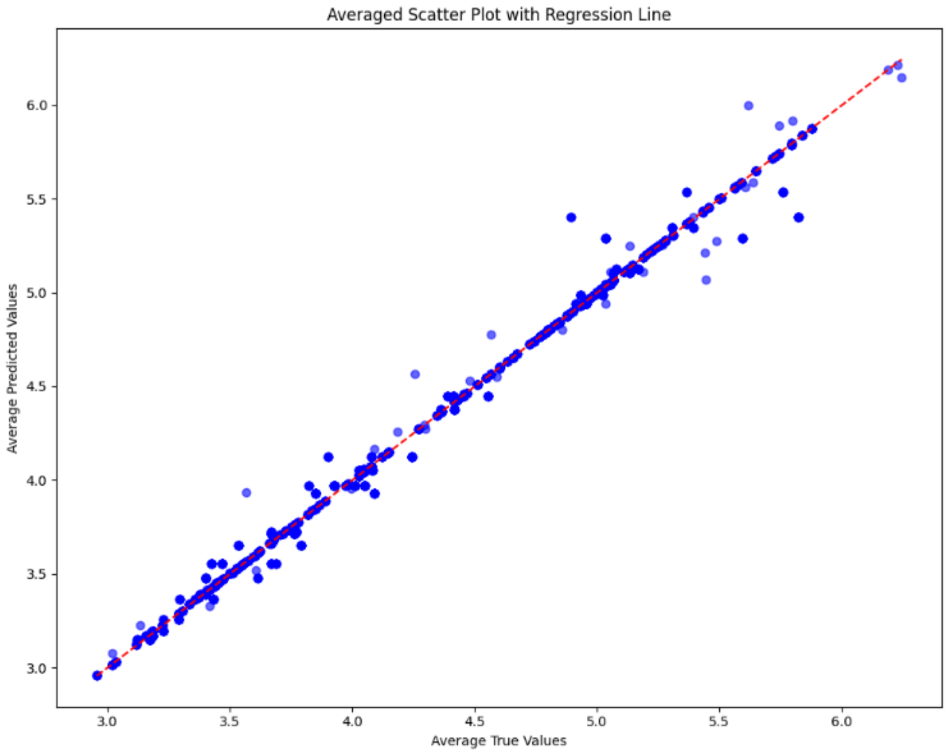



*Figure 6-10 - Averaged Scatter Plot with Best-Fit Line Created by XGBoost Regressor with the windows widths sub-data set*

Regarding predictions of windows existence in walls, several classifiers were trained including random forest, support vector classifier (SVC), k-NN, XGBoost, AdaBoost, and decision tree. The problem features a multi-class classification problem. To apply the algorithms, for each algorithm, 12 classifiers were trained so that each classifier is responsible for learning and predicting one class from the 12 targets. And to evaluate the model, average accuracies, recalls, precisions, and f1 scores are calculated for the 12 classifiers that each model has. All the 6 algorithms performed well with the data set and achieved high scores in all the metrics. Only SVC had slightly lower scores.

Figure 6-11 and table 6-4 show a comparison of the metrics achieved by the 6 algorithms.

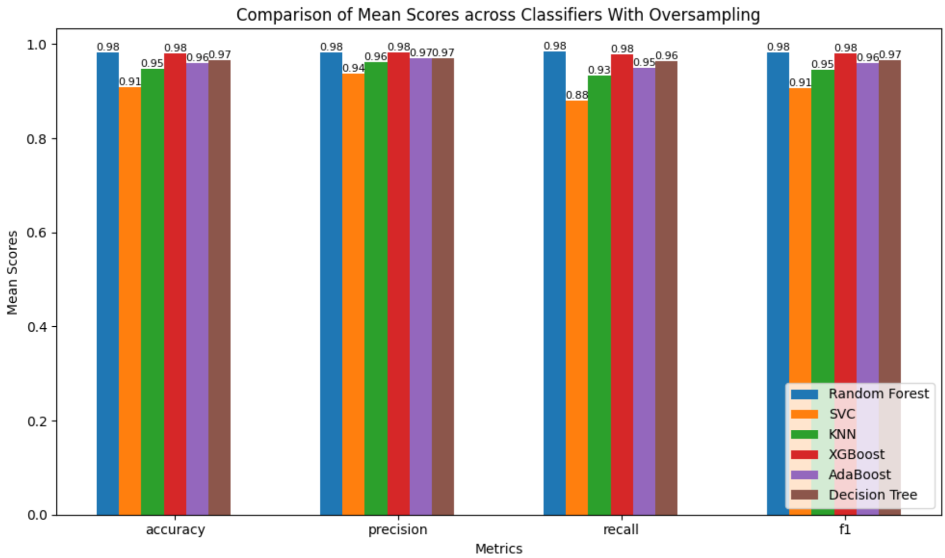



*Figure 6-11 - Comparison of the metrics achieved by the 6 classification algorithms trained with the windows existence sub-data set.*

*Table 6-4- Comparison of the classification metrics between the 6 trained models for the windows existence sub-data set*

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Random Forest | 0.98 | 0.98 | 0.98 | 0.98 |
| SVC | 0.91 | 0.94 | 0.88 | 0.91 |
| k-NN | 0.95 | 0.96 | 0.93 | 0.95 |
| XGBoost | 0.98 | 0.98 | 0.98 | 0.98 |
| AdaBoost | 0.96 | 0.97 | 0.95 | 0.96 |
| Decision Tree | 0.97 | 0.97 | 0.96 | 0.97 |

A confusion matrix is a tabular representation that illustrates the various outcomes arising from predictions and actual results in a classification problem. It provides a structured presentation of the classifier's predictions and the true values, aiding in the visualization of their interactions. The matrix displays a comprehensive overview of predicted and observed values within the classification process. Confusion matrices are visualized for all the classifiers in all classification models as shown in figure 6-12.

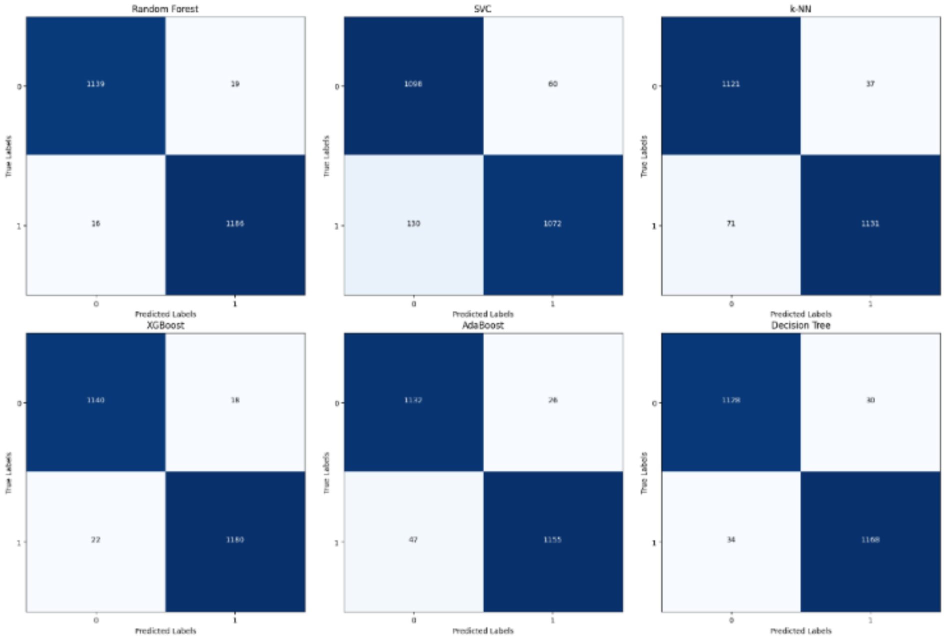



*Figure 6-12 - Confusion matrices for all the classifiers in all classification models*

The matrices show how all the models performed well and had very low classification errors compared to the right decisions. It also shows that SVC had the greatest number of false positive values and false negative values.

Figures 6-13 to 6-15 show the confusion matrices of the 12 classifiers of the random forest model which achieved the best overall results, XGBoost, and decision tree models respectively.

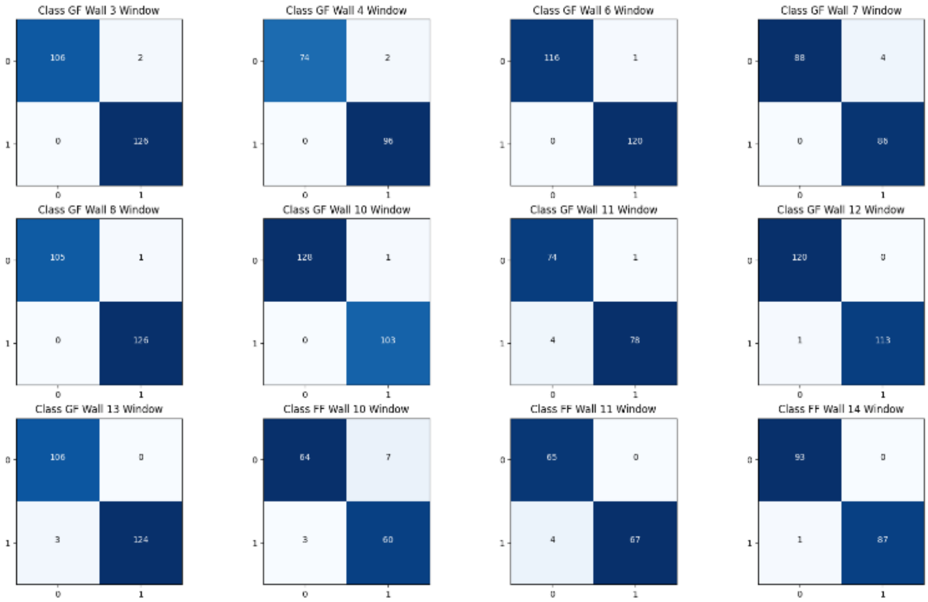



*Figure 6-13 - the confusion matrices of the 12 classifiers of the random forest model*

The matrix shows how the classifier with the largest number of errors (Class FF Wall 10 Window) had falsely predicted only 10 times out of 134 predictions.

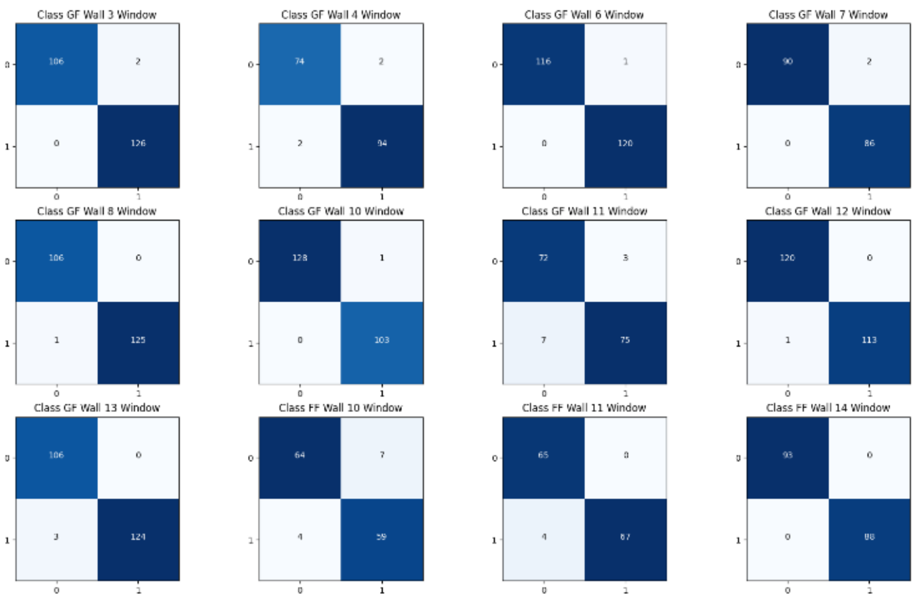



*Figure 6-14- the confusion matrices of the 12 classifiers of the XGBoost model*

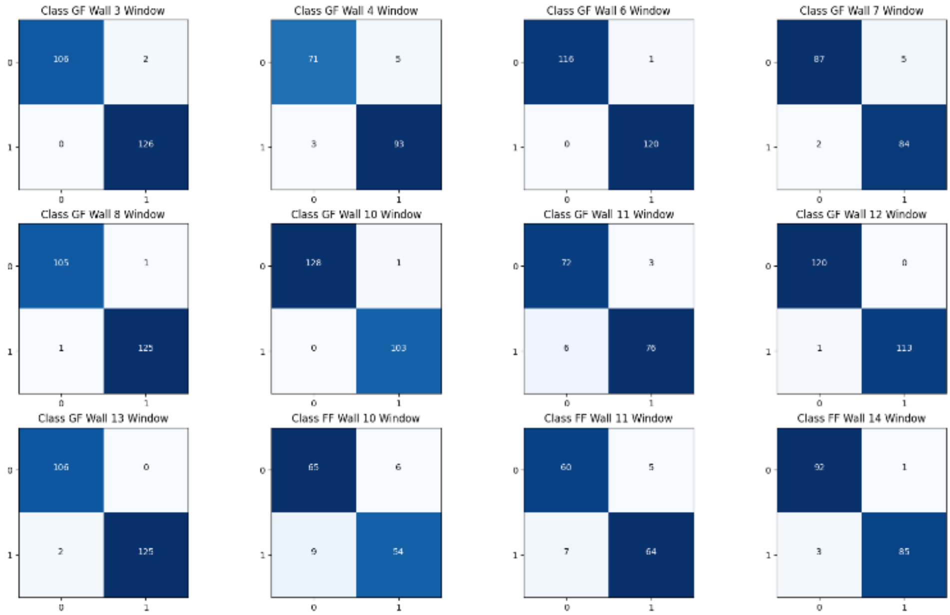



*Figure 6-15 - the confusion matrices of the 12 classifiers of the decision tree model*

Again, from the present results, it is significant that both the regressor and classifier algorithms performance were remarkable in both windows sub-data sets in terms of the evaluation metrics. Thus, many algorithms could generalize on unseen data.

## 6.4 Predictions

To sum up, the study was conducted on two architectural design data sets that are prepared to be numerical through coding. ML algorithms were trained on both data sets. Some regression models were trained on the first data set to predict the values of the slabs dimensions and other parameters related to walls dimensions and spacings. While other regression and classification algorithms were trained on the second data set to predict windows width and windows existence, respectively. However, to predict windows existence and windows widths, the same inputs passed for the 1st data set's predictions are not used because all the algorithms are required to complete a one single design prediction process. So, the passed inputs are the predictions of the slabs' dimensions obtained from training the first set of algorithms which are trained on the first data set as inputs to predict the windows existence and windows widths with the second set of algorithms. In this sense, one seamless connected framework is achieved utilizing ML in architectural form finding based on the architects' data and preferences which are used to train ML models.

### 6.5 Model Fine-Tuning

To make sure that the final design of the villa is satisfying, a fine-tuning phase is modeled to modify the design obtained by parsing ML output values. In this phase, the architect has all the control needed to increase/decrease any parameter to reach the required final form. Parameters are controlled with low values sliders to achieve a precise satisfying design. This resulting design, as well as any future designs could be easily added to the dataset and the ML cycle can be triggered again which should be enhanced when more designs are fed to it.

### 6.6 Discussion

The main aim of the present study was to evaluate the suggested ML framework where an architectural model is transformed into data sets containing all the possible parameters in a form of a numeric CSV files ready to train ML algorithms. Two data sets are applied: one related to slabs dimensions and walls dimensions and spacings and one related to windows existence and windows dimensions. In ML, data sets play a crucial role in the success of algorithms in generalizing for unseen data. So, an important part of this research was to pre-process the data and check if patterns can be found. In this regard, sub-data sets were created from the original data sets changing the targets and features for each one depending on the features importance to targets. Moreover, some targets related to slabs dimensions had to be changed so that the built-up area of the training data maps correctly with the slabs' dimensions and by trial, this proved to be important for the algorithm to create a logical best-fit line to predict the dimensions correctly. Also, some targets and features were consistent along all the samples, so they were dropped.

Training the models with cleaned data after the previous step was successful and the evaluation metrics were acceptable. However, another step was taken to optimize the algorithms performance which was oversampling. The data sets originally had 600 samples. The form data set was oversampled to have 30,600 samples by increasing the samples with minor classes, while SMOTE was used to oversample the windows classification sub-data set to balance the data. And this enhanced the algorithms' performance significantly.

After data engineering and oversampling, a set of ML algorithms were trained on the datasets including -for regression- ensemble learning methods like random forest, XGBoost, and decision tree, and other algorithms like linear regression, polynomial regression, ridge, k-NN, and Multi-layer perceptron feed forward NN. For classification, algorithms included random forest, XGBoost, decision tree, SVC, k-NN, and MLP.

Ensemble learning methods were very successful in the whole study. All of these algorithms succeeded in terms of achieving the best evaluation metrics among other algorithms.

As shown in table 6-5, overall, random forest regressor performance was very successful regarding evaluation metrics when trained with both of form's sub-data sets.

*Table 6-5 – Random Forest Metrics Scores for Roofs Data Set*

| Sub-data set | Best $R^2$ Score | Mean $R^2$ Score | MSE | Mean RMSE | MAE |
|---|---|---|---|---|---|
| **Areas** | 0.97 | 0.97 | 6.15 | 2.48 | 0.54 |
| **Rest of parameters** | 0.79 | 0.78 | 0.26 | 0.51 | 0.1 |

XGBoost, showed the best results when compared to other algorithms with the windows width sub-data set. Table 6-6 compares the results of the most successful algorithms in this scenario. The table shows how 5 out 7 algorithms performed very similarly to each other.

*Table 6-6 - Metrics comparison between different algorithms trained on the windows widths sub-data set*

| Algorithm | Best $R^2$ Score | Mean $R^2$ Score | MSE | Mean RMSE | MAE |
|---|---|---|---|---|---|
| **XGBoost** | 0.99 | 0.96 | 0.07 | 0.26 | 0.03 |
| **Random Forest** | 0.98 | 0.96 | 0.07 | 0.27 | 0.03 |
| **k-NN** | 0.99 | 0.97 | 0.08 | 0.28 | 0.03 |
| **Decision Tree** | 0.99 | 0.97 | 0.07 | 0.27 | 0.03 |
| **MLP** | - | 0.99 | 0.06 | 0.24 | 0.12 |

Even the classification task to predict whether a window exists or not, has seen a huge success regarding evaluation metrics. To do this task with the multi-class classification problem, 12 classifiers were created for each algorithm. Each classifier was trained to predict one output only. In this manner, MLP showed great results unlike when trained on continuous data. Still, ensemble learning methods showed better overall performance, especially the random forest algorithm for which a confusion matrix was visualized, and the number of false predictions was very low. Also, the k-NN algorithm performed exceptionally in this task.

The success of ensemble learning methods was expected as ensemble learning tends to combine the predictions of multiple base models, often leading to better overall predictive performance compared to individual models. This can result in higher accuracy, lower error rates, and improved generalization to new, unseen data. Also, by aggregating the predictions of multiple models, ensemble methods can help mitigate overfitting, which occurs when a model is too complex and performs well on the training data but poorly on the test data. Ensemble methods tend to make the final predictions more robust and less prone to overfitting. In addition, ensembles capture different aspects of intricate relationships through diverse base models, allowing for a more comprehensive understanding of the data.

To predict values to be parsed to the code on grasshopper3d to build the predicted model, predictions were taken first from the roofs and rest of parameters data. The predicted slabs dimensions were then used as inputs to predict windows existence and windows' dimensions. And the resulting model was very satisfying as it was built with the same architectural style that the models were trained with. The results were very similar to the visualized samples renders.

Subsequently, a fine-tuning stage was introduced within the framework to guarantee the development of a thoroughly refined product. During this phase, the architect gains complete control over all the parameters, albeit within a more constrained range of values, facilitating effortless model adjustments.

**Summary**

The present study is done to search for a proper framework for utilizing ML power in architectural design. To achieve this goal, various steps were taken to transform an architectural model into an ML ready dataset. The deep study of proper parametric relationships between the model's components was crucial to translate the components into the smallest possible unit of data representing coordinates, dimensions, and boolean options.

Using coding was very important to create such a complex network of interconnected parameters and to automate the tasks of exporting the parameters to a dataset in the form of a CSV file that is ready to be used in a ML pipeline. In a normal design and architectural modeling workflow, even using parametric tools and software, creating many designs/prototypes of the same design style takes a lot of time in modeling and modifying each prototype. This process could need even more architects depending on the number or required prototypes. But as the study proves, not only did coding facilitate how an architect can read geometry as a container of information, but also the creation of a great number of samples with the same design style leading to a large dataset in much less time than a usual design workflow can take to model different prototypes with different areas and parameters.

After that, the dataset was validated by extracting simpler dataset to predict the roofs dimensions to ensure that an ML regression model can predict parameters value that could lead to a building with a requested area. Exploratory data analysis was done to extract the variables that are most correlating to the roofs' dimensions. And data pre-processing was needed to ensure that the model can predict well by replacing some roofs dimensions with roofs areas with simple mathematical equations. This step was crucial for the success of the study because negative values of some roofs led to very low metrics scores when a regressor was trained. Also, data was oversampled to ensure better learning for the algorithm where the number of samples was increased by 30,000. After trials, this step proved to be very important for the success of the model which could not generalize well on 600 samples, a small sample space. The dataset was split into train/test sets with ratios of 80% and 20% respectively and a simple random forest regressor was trained on the dataset with its default parameters.

The study exhibited a great success with the random forest regressor for which the scores metrics $R^2$ score, MSE, RMSE, and MAE were 0.97, 6.15, 2.48, and 0.54, respectively. A practical validation was followed by reversing the mathematical calculations to get the villa's area which happened to be the same as the required area or very close to it. The predicted parameters were then parsed into the code of the villa model with only a button click in grasshopper canvas to check how the model looks like. What this study proved is that the machine could learn and map the patterns that an architect follows when designing a building. It is like the architect taught the machine how he designs. A process that is hard to explain to another architect in words. This approach could lead to a great effect in the architectural design process especially that the result is as close as possible to what the architect could think inside his brain.

Finally, a fine-tuning phase was added to the framework to ensure a satisficing product for the architect. In this phase, the architect gets full control of all the parameters but with a smaller range of numbers to modify the model easily.

The results can be described in terms of time and effort distributed between AI and human interventions across two stages of a design process shown in figures 6-16 to 6-18. In Stage 1, the pie charts illustrate the time allocation between AI (30%) and human input (70%) in the design process, with human intervention being dominant. On the right, a parallel comparison for coding shows a similar distribution, where design comprises 30%, and coding consumes 70% of the effort.

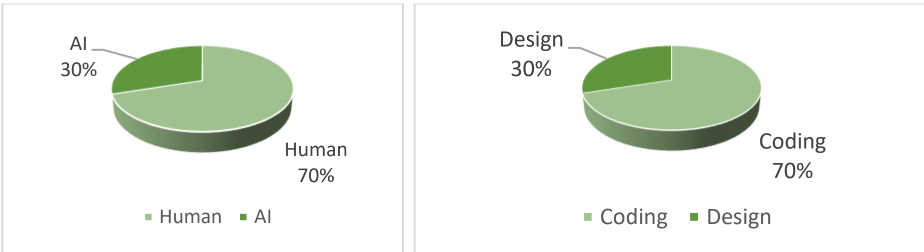



*Figure 6-16 Stage 1: Time and Effort Estimation for AI/Human (left) and human design/coding (right)*

Stage 2 shifts significantly towards AI involvement, with AI contributing 95% and human involvement reduced to 5%, indicating an advanced level of automation.

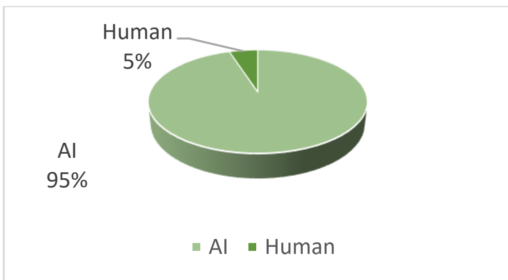



*Figure 6-17- Stage 2: Time and Effort Estimation for AI/Human Intervention*

The overall assessment at the bottom, showing AI taking 30% and human 70%, summarizes the cumulative effort across both stages. This demonstrates the evolving role of AI in design and coding, highlighting the efficiency gained from AI while retaining essential human input in the creative stages.

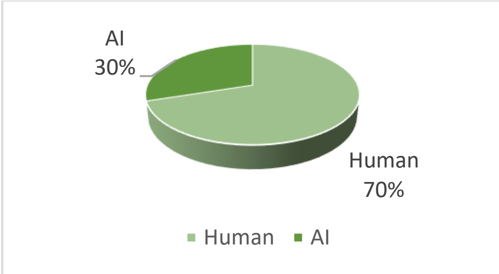



*Figure 6-18- Overall: Design Intervention Estimation*

Regarding time, the estimated time for stage 1, which includes modelling the villa using coding, creating 600 consciously designed samples, and training ML models took around 15 days while stage 2 takes a maximum of 5 minutes to generate a new design.

# Research Conclusion

The present study provides evidence about the efficiency of utilizing ML algorithms with architectural design numeric data sets and coding being an important skill for architects. The study proves a novel direction on how to apply ML in the form generation process.

Artificial Intelligence has seen a lot of improvements and applications recently in many fields and architecture is no exception. However, being a field which exhibits a lot of creativity and logic makes it hard to apply AI algorithms as with other fields. Architecture is considered a complicated practice which requires knowledge in many disciplines including psychology, physics, mathematics, art, and more. The product of architecture is considered to be complicated as well because of the many considerations that accompany it. An architect thinks in a cyclic way through a complex design process to produce a building design. This makes the learning of architectural and creative thinking as well as problem solving an essential necessity to architects. Add to that, how at some point an architect takes decisions based on a black box thinking approach. In this case, can the machine and today's AI algorithms map how the architect thinks?

According to the analytic study of architectural design elements and motifs, it is clearly observed that those elements and motifs are repeatedly used by architects according to the design era. A pattern between them could be identified. For instance, in contemporary architecture of villas, the use of L, C, U shapes in facades by modifying slabs and walls to create frames is observed as a repetitive motif. Additionally, the use of rectilinear shading elements is repeatedly used by architects. If one could observe such patterns, the machines can also do so, today.

Design methods including parametric design, generative design, algorithmic design, etc. benefited from what the computers could do, adding an algorithmic characteristic to the design process making it clearer and more logical to the architect. Even some decisions regarding structural, environmental, and energy aspects could be taken using optimization. All of these methods would not have seen light without the advent of visual programming languages where an architect codes the building using parameters and operations on them. Moreover, coding is considered to have a significant effect on the design process giving the architect more power, freedom, and spruceness as well as helping them break free from any bias that is present in today's modelling software and educational phase.

AI applications in architecture have seen many trials since the late 19$^{th}$ century. However, they became more popular when image generation through Gen-AI models was introduced. In essence, AI algorithms could do much better than creating images which is believed to be machine-centric today rather than human-centric because of the image generation process and the data the algorithm learns with. AI algorithms could help in automating many tasks including predicting numbers, classifying numbers and images, clustering, etc. Such algorithms are ensured to make the design process more human-centric where the architect feeds the algorithm with dataset related only to his problem while the dataset being designed by the architect themselves based on their experience or past projects. These benefits of using non-gen AI against the image generation tasks which cannot map a real complicated design process dealing with the building as information drawn from a lot of issues and disciplines as well as authenticity issues, show non-gen AI's superiority.

To utilize machine learning in the 3d model generation (form finding), transforming architects' ideas and designs into numbers become essential and could be achieved by dealing with the buildings' parameters as small entities of data. This could show a huge advancement in applying ML in the architectural design field to automate tasks that consume a lot of time like creating many prototypes with different parameters but with the same design style. Coding could open many possibilities by altering the architect's mind from thinking about geometry as geometry to thinking about it as a container of information. The information could create an infinite number of possibilities regarding how buildings' components correlate with each other parametrically. This approach could utilize automating the creation of a dataset of architectural design parameters to be used to train ML models which could predict and automate the design phase of new prototypes of the same design style with different parameters and properties.

To overcome the data set creation challenge and generate hundreds of studied designs in a short time while obtaining valuable insights from the data using ML techniques, an architectural 3d model was generated parametrically so, its parameters should be strongly related and could be transformed into data sets containing all the possible parameters in a form of a numeric CSV file ready to train ML algorithms. For this task, the model was algorithmically designed and coded in C# using RhinoCommon geometry functions in a sense where every parameter was related to a target such as the total built-up area, land dimensions, neighbor types, and setbacks. Two datasets were generated. A form dataset is designed with parameters related to the building design including walls lengths, slabs lengths and widths, heights, number of building blocks, louvers' numbers and distances between them, as well as walls distance from slabs. The other data set is designed to have windows data including window's existence in each wall and their widths.

In this study, six hundred samples of designs with 122 parameters were created. The inputs (features) were the length and width of the land, neighbors, built-up area, setbacks, and a starting point. On the other hand, the outputs (targets) were all the numeric and textual parameters of the walls and windows (112 parameter).

Training ML algorithms with the two data sets exhibited some challenges, including weak relations between some parameters and data imbalance due to design limitations. To address these issues, the data sets had to be preprocessed and engineered to ensure that the relationships between different parameters are clearer to the machine. Some string parameters were transformed into numeric values, and some parameters were mathematically processed.

To tackle creating a larger data set while balancing it, another step was taken to optimize the algorithms' performance which was oversampling. The data sets originally had six hundred samples. The form data set was oversampled to have thirty thousand and six hundred samples by increasing the samples with minor classes, while SMOTE was used to oversample the windows classification sub-data set to balance the data. This enhanced the algorithms' performance significantly especially when ensemble learning algorithms were applied. After data engineering and oversampling,

After preprocessing the data and splitting the data sets to training and test data sets using K-cross validation, a set of ML algorithms were trained on the datasets including -for regression- ensemble learning methods like random forest, XGBoost, and decision tree, and other algorithms like linear regression, polynomial regression, ridge, k-NN, and MLP. On the other hand, for classification, algorithms included random forest, XGBoost, decision tree, SVC, and k-NN. To solve the multi-class classification problem, twelve classifiers for each algorithm were created. Each classifier was trained to predict one output only, which was whether a window exists or not. The models were evaluated using regression metrics like R2 score, MSE, RMSE, and MAE, and classification metrics like accuracy, precision, recall, and F1.

Overall, in regression tasks, random forest regressor performance was successful regarding evaluation metrics when trained with both of form's sub-data sets. Additionally, XGBoost showed the best results when compared to other algorithms with the windows width sub-data set. Additionally, the classification task was successful in terms of evaluation metrics. In this manner, ensemble learning methods showed better overall performance, especially the random forest algorithm for which a confusion matrix was visualized, and the number of false predictions was low. Also, the k-NN algorithm performed exceptionally in this task. The success of ensemble learning methods was expected because these models tend to combine the predictions of multiple base models, often leading to better overall predictive performance compared to individual models. In addition, ensembles capture different aspects of intricate relationships through diverse base models, allowing for a more comprehensive understanding of the data. Also, by aggregating the predictions of multiple models, ensemble methods could help mitigate overfitting, which occurs when a model was too complex and performs well on the training data but poorly on the test data. This resulted in higher accuracy, lower error rates, and improved generalization for new, unseen data.

To predict values which were parsed to the code on grasshopper3d to build the predicted model, predictions were taken first from the roofs and rest of parameters data. The predicted slabs dimensions were then used as inputs to predict windows existence and windows' dimensions. As practical validation, the predictions were validated numerically by making sure the design achieves the required area and respects legal constraints. Also, it was an easy task to tell if the predicted numbers lead to a satisfactory output because the training dataset was generated based on the authors' designs. The resulting model was satisfying as it was built with the same architectural style that the models were trained with and the required areas were predicted precisely. And the results were similar to the visualized samples renders. Subsequently, a fine-tuning stage was introduced within the framework to guarantee the development of a thoroughly refined product. During this phase, the architect gains complete control over all the parameters, albeit within a more constrained range of values, facilitating effortless model adjustments. Generally, the algorithms were successful because the designed data set already ensured clear relationships between targets and features. And this proves that architectural design is based on traceable rules applied to the design algorithms by the designer. In this case, the machine could automate the 3D model design process by learning these rules and predicting based on them.

Finally, the suggested framework is tested against tangible aspects of architectural design which do not depend on certain design patterns but adapt to the architect's decisions presented in the final parameters of the data set. Additionally, only straightforward form aspects related to proportions were tested for simplicity. Adding more aspects to the design such as environmental aspects could make the patterns harder to find by the ML algorithms. However, if patterns exist between the features and targets -building requirements and building parameters- the framework should lead to 'accurate' results and in this case accurate denotes the architect's way of thinking showing in the parameters they choose and create.

## Future Directions

As advancements in artificial intelligence and ML are already affecting all the fields daily, architectural design field is no exception. Explorations with how to automate architectural design tasks and applying ML to the design process are becoming inevitable, especially that the architectural field is considered one of the latest fields to benefit from AI.

Future research could include how to improve the data sets created by coding because data is the most important aspect when applying ML in any field. How to increase the number of samples is a crucial requirement. Increasing the number of samples with varying parameters space could lead to better trained ML models. For instance, adding more samples with much varying land lengths and widths, built-up areas, etc., with smaller differences range could enhance training the models.

Also, creating different sets regarding architectural design style with more parameters and more options to train a model to predict the parameters of the building according to its style, wide range of heights, number of blocks, and typologies could be efficient to generalize the prospect applications leading to better performance on ground.

Future investigations could also consider creating models on other design platforms away from Rhinoceros3d and Grasshopper3d. Investigations on creating a whole new type of software that automates the coding process of the data sets, especially with new AI coding Copilots introduced lately as well as Large Language Models which deal with coding like GPT and Llama is essential. The new software could then use ML algorithms to train on the data sets created and return the architectural form instantly. Such applications could lead to a much lighter and easier to use interface that makes the 3D modelling process smoother and more straightforward.

Additionally, exploring more advanced and intricate design parameters could lead to a design framework that is much closer to the real world.

Such parameters could include various design aspects like environmental, cost, as well as other intangible aspects such as psychological and philosophical aspects of architecture. For instance, adding features (building requirements) to the design samples including daylighting metrics such as sDA and ASE or solar radiation analysis results being independent parameters that affect different building parameters (targets) could lead to better results regarding an all-in-one architectural design form prediction approach. In this case, the architect could add intended sDA and ASE values to the input parameters to decide the target parameters of the building.

Other applications including materials choice could be investigated within the same framework to give more information regarding the resulting 3D model which could then be used within a BIM workflow seamlessly. Additionally, other building parameters could be explored including building orientation, rooms and services locations, interior design parameters, etc.

Also, other types of problems could be explored. For example, instead of training ML models to predict form parameters they could be trained to predict floor plans designs, urban design compositions, etc. Such applications could use the exact same framework, and more than one application could be integrated to work on different problems at the same time moving forward to a multi-tasking AI model. However, each problem will require more investigations regarding the best working ML models and ensemble learning models do not guarantee the result. In this case, deep learning models may capture more intricate patterns and suit better such complicated workflows.

Finally, enhancing the framework to be designed and used by multi-users needs investigations. If different users could generate design samples according to a certain goal, the framework could significantly enhance the design process. For example, a user could be responsible for choosing parameters regarding form proportions, another for specifying materials and cost, another for optimization tasks and environmental responsiveness, etc.

# References

## *Books, Papers, and Articles:*


1- Aalaei, M., Saadi, M., Rahbar, M., and Ekhlassi, A., 2023, Architectural Layout Generation Using a Graph-Constrained Conditional Generative Adversarial Network (GAN), *Automation in Construction 155 (2023) 105053*

2- Aamir, N., 2017, The Ideological Basis of Design Motifs in the Architecture of Subcontinent: The Converging Stories of Bells and Tassels, Pakistan Social Sciences Review

3- Abowardah, E., and Osama, M., 2016, Design Process and Strategic Thinking in Architecture, Proceedings of 2016 International Conference on Architecture & Civil Engineering (ICASCE 2016), London.

4- Aczel, D., 1989, Complete Business Statistics. Irwin, p. 1056. ISBN 0-256-05710-8.

5- Agkathidis, A., 2015, Generative Design: Form-Finding Techniques in Architecture, Laurence King Publishing Ltd, UK., pp. 14-17.

6- Aggarwal, C., 2015. Data mining: The textbook. Springer.

7- Ahlquist, S., and Menges, A., 2011. Introduction: Computational Design Thinking. In: Menges, A., Ahlquist, S. (Eds.), *AD Reader: Computational Design Thinking*. John Wiley & Sons Ltd, United Kingdom, pp. 10e29.

8- Akizuki, Y., Bernhard, M., Kakooee, R., Kladeftira, M., and Dillenburger, B., 2020, Generative Modelling with Design

Constraints – Reinforcement Learning for Object Generation – *CAADRIA Volume 1* - Pages / Article No. 445 – 454

9- Ali, S., 2020, The Role of Visualization Software in Architectural Education مجلة العمارة والفنون والعلوم الإنسانية *Article 3, Volume 5, Issue 24*, November 2020, Page 26-43

10- Alpaydin, E., 2014, Introduction to Machine Learning, The MIT Press, Cambridge, Massachusetts, London, England.

11- Alymani, A., Jabi, W., and Corcoran, P., 2020, Machine Learning Methods for Clustering Architectural Precedents Classifying the relationship between building and ground, eCAADe 38.

12- Amamata, Z., Mohammed, B., and Yusuf, B., 2014, Are Smart Buildings Same as Green Certified Buildings? A Comparative Analysis, Int. J. Sci. Res. Publ. 4 2250– 3153 www.ijsrp.org.

13- Archer, L.B., 1969, The Structure of the Design Process. In G. Broadbent & A.Ward (Eds.) *Design Methods of Architecture*. New York: Witteborn.

14- As, I., Pal, S., and Basu, P., 2018, Artificial Intelligence in Architecture: Generating Conceptual Design via Deep Learning, *International Journal of Architectural Computing, Vol. 16(4)*, pp. 306-327.

15- As., I., Pal, S., and Basu, P., 2023, Artificial Intelligence in Architecture: Generating Conceptual Design Via Deep Learning, *International Journal of Architectural Computing 2018, Vol. 16(4)* 306–327.

16- Asimow, M. 1962, Introduction to Design, Prentice-Hall.

17- Atlas, L., Park, D., El-Sharkawi, M., Marks, R., and Damborg, M., 2002, Electric Load Forecasting Using an Artificial Neural

Network, *IEEE Trans. Power Syst. 6* 442–449, doi:10.1109/59.76685.

18- Bacon, E., 1974, Design of Cities, Penguin Publishing Group.

19- Banham, R., 1996, A Critic Writes: Selected Essays by Reyner. Banham. Berkeley: University of California Press, c1996.

20- Banko, M., & Brill, E. (2001). Scaling to Very Large Corpora for Natural Language Disambiguation. Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (pp. 26-33).

21- Bao, Y and Xiang, C., 2023 - Exploration of Conceptual Design Generation based on the Deep Learning Model-Discussing the Application of AI Generator to the Preliminary Architectural Design Process, xArch symposium-Creativity in the age of digital reproduction At: Xi'an Jiaotong-Liverpool University, Suzhou, China

22- Basarir, L., and Erol, K., 2021, Briefing AI: From Architectural Design Brief Texts to Architectural Design Sketches ASCAAD 2021

23- Belém, C, Santos, L and Leitão, A 2019 'On the Impact of Machine Learning Architecture without Architects?', *CAAD Futures 2019*, Daejon, South Korea

24- Bellman, R. (1987). An Introduction to Artificial intelligence: Can computers think? Thomson Course Technology.

25- Bengesi, S., El-Sayed, H., Sarker, M., Houkpati, Y., Irungu, J., and Oladunni, T., 2023, Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers, arXiv:2311.10242 [cs.LG]

26- Bergstra, J., & Bengio, Y., 2012, Random Search for Hyper-Parameter Optimization, *Journal of Machine Learning Research*, 281–305. http://dx.doi.org/10.5555/2188385.2188395

27- Bloch, T., and Sacks, R., 2018, Comparing Machine Learning and Rule-Based Inferencing for Semantic Enrichment of BIM Models, *Automation in Construction* 256-272

28- Bovill, C.: Fractal Geometry in Architecture and Design. Birkhauser, Boston, 1996.

29- Boukarta, S., 2021, Predicting energy demand of residential buildings: A Linear Regression-Based Approach for A Small Sample Size. SSP – *Journal of Civil Engineering, Vol 16*, Issue 2/

30- Breiman, L., 1984, Classification and Regression Trees, 1st Edition, Routledge, New York.

31- Breiman, L. ,1996. Bagging Predictors. *Machine Learning, 24(2)*, 123–140.

32- Breiman, L. ,1998. Arcing Classifiers. *The Annals of Statistics, 26(3)*, 801–849.

33- Breiman, L., 2001a. Statistical Modeling: The Two Cultures. *Statistical Science, 16,* 199-215.

34- Breiman, L., 2001b. Random Forests. *Machine Learning, 45*, 5-32.

35- Bishop, C. M. 2006, Pattern Recognition and Machine Learning. Springer.

36- Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S., Fidler, S., and Kreis, K., 2023, Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models, arXiv:2304.08818v1 [cs.CV]

37- Broadbent, G., 1969, Design Methods in Architecture, New York, G. Wittenborn

38- Broadbent, G., 1973. Design In Architecture: Architecture and the Human Sciences. UK: John Wiley and Sons.

39- Broadbent, G., (postscript), (1988). Design in Architecture: Architecture and the Human Sciences. London: David Fulton Publishers.

40- Burry, J. & Burry M., 2010, The New Mathematics of Architecture, Thames & Hudson Ltd, UK.

41- Burry, M. (2016). Prototyping for architects. Thames & Hudson.

42- Bukhari, F. A, 2011. A Hierarchical Evolutionary Algorithmic Design (HEAD) System for Generating and Evolving Building Design Models. PhD Thesis. Queensland University of Technology.

43- Caetano, I., Santos, L., Leitao, A., 2019, Computational Design in Architecture: Defining Parametric, Generative, and Algorithmic Design

44- Caldas, L., 2008. Generation of Energy-Efficient Architecture Solutions Applying GENE_ARCH: An Evolution-Based Generative Design System. Adv. Eng. Inf. 22, 59e70.

45- Caliskan, E., 2023, Interview with Chat GPT to Define Architectural Design Studio Work: Possibilities, Conflicts and Limits, Journal of Design Studiov:5

46- Capon, David Smith, (1999a). Architectural Theory, Volume One: The Vitruvian Fallacy, A History of the Categories in Architecture and Philosophy. West Sussex: John Waley & Sons.

47- Capon, D., l999b, Architectural Theory, Volume Two: Le Corbusier's Legacy, Principles of Twentieth-century Architectural Theory Arranged by Category. West Sussex: John Waley & Sons.

48- Ceylan, D., et. Al, 2023, Pix2Video: Video Editing using Image Diffusion, arXiv:2303.12688v1 [cs.CV] 22 Mar 2023

49- Cerquitelli, T., Malnati, G., and Apiletti, D., (2019). Exploiting Scalable Machine-Learning Distributed Frameworks to Forecast Power Consumption of Buildings. Energies 12 (15) pp. 29-33.

50- Chae, Y., Horesh, R., Hwang, Y., and Lee, Y., 2016, Artificial Neural Network Model For Forecasting Sub-Hourly Electricity Usage In Commercial Buildings, *Energy Build 111* 184–194, doi:10.1016/j.enbuild.2015.11.045.

51- Chai, W., Guo, X., Wang., G., and Lu, Y., 2023, StableVideo: Text-Driven Consistency-aware Diffusion Video Editing, arXiv:2308.09592v1 [cs.CV] 18 Aug 2023

52- Chaillous, S., 2019, AI & Architecture – An Experimental Perspective – Harvard University

53- Chandrasegaran, E., Kisselburgh, L., & Ramani, T. (2012). Enhancing visual thinking in a toy design course using freehand sketching. August Proceedings of the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE.

54- Charniak, E., & McDermott, D. V. (1985). Introduction to Artificial Intelligence. Addison-Wesley.

55- Charytoniuk, W., and Chen, M., 1998, Nonparametric Regression Based Short-Term Load Forecasting, IEEE Trans. Power Syst. 13 725–730, doi:10.1109/59.708572.

56- Chou, J., and Bui, D., 2014, Modeling Heating And Cooling Loads By Artificial Intelligence For Energy-Efficient Building Design, Energy Build 82 437–446, doi:10.1016/j.enbuild.2014.07.036.

57- Chawla, N., Bowyer, K., Hall, L., and Kegelmeyer, W., 2002, SMOTE: Synthetic Minority Over-sampling Technique, Artificial Intelligence Research (JAIR)

58- Ching, F.D.K. (1996) Architecture; Form, Space & Order. Van Nostrand Reinhold, New York.

59- Clement, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., & Stafford, J. (2011). Documenting software architectures: Views and beyond (2nd ed.). Addison-Wesley.

60- Conway, H., and Roenisch, R., 1994, Understanding Architecture: An Introduction to Architecture and Architectural History, Francis & Taylor e-Library, 2005.

61- Cross, N., 2006, Designerly ways of Knowing. London: Springer-Verlag.

62- Cudzik, J., and Radziszewski, K., 2018, Artificial Intelligence Aided Architectural Design, *AI FOR DESIGN AND BUILT ENVIRONMENT - Volume 1 - eCAADe* 36 | 77-84

63- Culter, L., and Culter, S., 1982. Recycling Cities for People: The Urban Design Process. 2nd ed. Boston, Mass.: CBNI Publishing.

64- Dedinec, A., Filiposka, S., Kocarev, L., 2016, Deep Belief Network Based Electricity Load Forecasting: An Analysis Of Macedonian Case, *Energy 115* 1688–1700, doi:10.1016/j.energy.2016.07.090.

65- Deng, H., Fannon, D., and Eckelman, M., 2018, Predictive Modeling For US Commercial Building Energy Use: A Comparison Of Existing Statistical And Machine Learning Algorithms Using CBECS Microdata, Energy Build 163 34–43, doi:10.1016/j.enbuild.2017.12.031.

66- Desai, N., & Patel, V., 2021, Linear Decision Tree Regressor: Decision Tree Regressor Combined with Linear Regressor.

67- Dong, B., Cao, C., and Lee, S., 2005, Applying Support Vector Machines To Predict Building Energy Consumption In Tropical

Region, *Energy Build 37* 545–553, doi:10.1016/j.enbuild.2004.09.009.

68- Dorst, K., 2003, Exploring the Structure of Design Problems, International Conference on Engineering Design Iced 03 Stockholm.

69- Duarte, J. P. (2001). Customizing mass housing: A discursive grammar for Siza's Malagueira houses. MIT Press.

70- Duerk, Donna P., (1993). Architectural Programming: Information Management for Design. New York: Van Nostrand Reinhold.

71- Edwards. R., New, J., and Parker, L., 2012, Predicting Future Hourly Residential Electrical Consumption: A Machine Learning Case Study, Energy Build 49 591–603, doi:10.1016/j.enbuild.2012.03.010.

72- Eggert, R.J., 2004, Engineering Design, Prentice Hall, Upper Saddle River, New Jersey.

73- Elghandour, A., Saleh, A., Aboeineen, O., and Elmokadem, A., 2014, Using Parametric Design to Optimize Building’s Façade Skin to Improve Indoor Daylighting Performance. pp. 353-361

74- El Iraqi, A., and El Daly, H., 1997, Architectural Forming Between Form Making and Form Finding Towards Form Follows Parameters, Journal of Al-Azhar University Engineering Sector

75- Erzen, N., 2015, Form and Meaning in Architectural Theory, *Serbian Architectural Journal 7* (2), pp.75-87

76- Fan, C., Xiao, F., and Zhao, Y., 2017, A Short-Term Building Cooling Load Prediction Method Using Deep Learning Algorithms, *Appl. Energy. 195* 222–233, doi:10.1016/j.apenergy.2017.03.064.

77- Ferriss, H., 1929, The Metropolis of Tomorrow, David McKay Publications, USA.

78- Fischer, T., Herr, C.M., 2001. Teaching generative design. In: Proceedings of the 4th International Conference on Generative Art. Milan, Italy.

79- Frazer, John, Frazer, Julia, Xiyu, L., Mingxi, T., Janssen, P., 2002. Generative and evolutionary techniques for building envelope design. In: Proceedings of the 5th International Conference on Generative Art. Milan, Italy, pp. 1e16.

80- Georges, M., (1954). L'architecture musulmane d'Occident. Paris: Arts et métiers graphiques. pp. 163–164.

81- Gero, J. S., & Kannengiesser, U. (2014). The function-behaviour-structure ontology of design. In A. Chakrabarti & L. T. M. Blessing (Eds.), An Anthology of Theories and Models of Design (pp. 263–283). Springer.

82- Ghani, H., Virdee, B., & Salekzamankhani, S., 2023, A Deep Learning Approach for Network Intrusion Detection Using a Small Features Vector, *Journal of Cybersecurity and Privacy*, pp. 451-463.

83- Gilhooly, K., J., 1996, Thinking: Directed, Undirected, and Creative, Elsevier Science.

84- Gonzalez, P., and Zamarreno, J., 2005, Prediction Of Hourly Energy Consumption In Buildings Based On A Feedback Artificial Neural Network, *Energy Build 37* 595–601, doi:10.1016/j.enbuild.2004.09.006.

85- Grant, P., (1975). "Aims and Potentials of Design Methodology". In: Basil Honikman ed. Responding to Social Change. Stroudsburg, Pa.: Hutchinson and Ross, 1975, pp. 96-108.

86- Grant, P., (1982). Design by Objectives: Multiple Objective Design Analysis and Evaluation in Architectural, Environmental and Product Design. San Luis Obispo, Ca.: Design Methods Group.

87- Green, R., (1962). The Architect's Guide to Running a Job. London: Architectural Press.

88- Gugelot, H., (1963). "Industrial Design in Practice". Zeitschrift der Hochschule fur Gestaltung.

89- Gungama, M., (2017). Generative Algorithms in Alternative Design Exploration, SHS Web of Conferences 41, 05003.

90- Haury, A., Gestraud, P., & Vert, J., 2011, The Influence of Feature Selection Methods on Accuracy, Stability and Interpretability of Molecular Signatures. *PLoS ONE, 6(12)*, e28210. http://dx.doi.org/10.1371/journal.pone.0028210.

91- Haugeland, J. (1985). Artificial intelligence: The very idea. MIT Press.

92- Hastie, T., Tibshirani, R., & Friedman, J. (2001). The Elements of Statistical Learning: Data Mining, Inference, And Prediction. Springer.

93- Hays, J., & Efros, A. A. (2007). Scene Completion Using Millions Of Photographs. ACM Transactions on Graphics (TOG), 26(3), 4.

94- He, Y., Liu, X., Zhang, H., Zheng, W., Zhao, F., Schnabel, M., and Mei, Y., 2021, Hybrid Framework for Rapid Evaluation of Wind Environment Around Buildings Through Parametric Design, CFD Simulation, *Image Processing and Machine Learning, Sustainable Cities and Society 73* 103092

95- Hechenbichler., K. & Schliep, K., 2004, Weighted K-Nearest-Neighbor Techniques and Ordinal Classification. Institute Für Statistik, Sonderforschungsbereich 386, Paper 399, p. 1-16

96- Hecht-Nielsen, R., 1990. Neurocomputing. Addison-Wesley, Reading, MA.

97- Herr, C., López, R., & Serrentino, R. (2002). Generative Architectural Design and Complexity Theory.

98- Heo, Y., and Zavala, V., 2012, Gaussian Process Modeling For Measurement And Verification of Building Energy Savings, Energy Build 53 7–18, doi:10.1016/j.enbuild.2012.06.024.

99- Hill, V., and Rowlands, P., 2008, Nature's Code, AIP Conf. Proc. 1051, 117–126 (2008)

100- Hillier, B., Musgrove, J. and O'Sullivan, P. (1972). Knowledge and design. In *Environmental Design: Research and Practice EDRA 3*. University of California.

101- Hong, G., Choi, G., Eum, J., Lee, H., and Kim, D., 2022, The Hourly Energy Consumption Prediction by KNN for Buildings in Community Buildings. *Buildings 12(10)* 1636.

102- Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W., 2021, LoRA: Low-Rank Adaptation of Large Language Models arXiv:2106.09685 [cs.CL] https://doi.org/10.48550/arXiv.2106.09685

103- Jabi, W., 2013, Parametric Design for Architecture, Lauerence King Publishing Ltd, London.

104- Janssen, P., Stouffs, R., 2015. Types of parametric modelling. In: Proceedings of the 20th International Conference of

Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2015, pp. 157e166.

105- Jones, J., Christopher, 1980, Design Methods, Wiley.

106- Kalay, Y.E., 1989. Modelling Objects and Environments (Principles of Computer Aided Design). Wiley-Academy, New York.

107- Karadag, I., Guzelci, O., and Alacam, S., 2022, EDU-AI: A Twofold Machine Learning Model to Support Classroom Layout Generation) *Construction Innovation Vol. 23 No. 4*, 2023 pp. 898-914 Emerald Publishing Limited

108- Karras, J., Holynski, A., Wang, T., and Kemelmacher-Shilzerman, I., 2023, DreamPose: Fashion Image-to-Video Synthesis via Stable Diffusion - arXiv:2304.06025v1 [cs.CV] 12 Apr 2023

109- Kesavaraj G. & Sukumaran S., 2013, A Study on Classification Techniques in Data Mining. In Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on, pp. 1-7.

110- Khabazi, Z., 2012, Generative Algorithms (Using Grasshopper), Digital Publication.

111- Knight, T.W, 2000, Shape Grammars in Education and Practice: History and Prospects. Online paper, Department of Architecture, MIT

112- Kolarevic, B., 2003, Architecture in the Digital Age - Design and Manufacturing, Spon Press, UK.

113- Kooa, B., La, S., Cho, N., and Yu, Y., 2019, Using Support Vector Machines to Classify Building Elements for Checking the

Semantic Integrity of Building Information Models, *Automation in Construction 98* 183–194

114- Kozlowski, T., 2013, The Beginnings of Contemporary Expressionism, *Technical Transactions, Vol. 110*.

115- Krause, J., 2003. Reflections: the creative process of generative design in architecture. In: Proceedings of the 6th International Conference on Generative Art.

116- Krawczyk, Robert J., 2002, Architectural Interpretation of Cellular Automata. Illinois Institute of Technology, College of Architecture, USA, Generative Art

117- Kristiansen, T., Jamil, F., Hameed, I., and Hamdy, M., 2022, Predicting Annual Illuminance and Operative Temperature in Residential Buildings Using Artificial Neural Networks, *Building and Environment 217* 109031

118- Kuang, Z., Zhang, J., Huang, Y., and Li, Y., 2023, Advancing Urban Renewal: An Automated Approach to Generating Historical Arcade Facades with Stable Diffusion Models - https://doi.org/10.48550/arXiv.2311.11590

119- Kurzweil, R. (1990). The Age of Intelligent Machines. MIT Press.

120- Lakzaeian, F., 2020, Point Cloud Segmentation and Classification of Structural Elements in Multiplanar Masonry Building Facades, *Automation in Construction 118* 103232

121- Latour, B., 1987, Science in Action – How to Follow Scientists and Engineers through Society – Harvard University Press -Massachusetts.

122- Lawson, B., 1980, Creative Thinking, In Book: *The Study of Real Skills Vol. 2*, MTP Press..

123- Lawson, B., 2005, How Designers Think – 4th Edition, Routledge, London, p.129

124- Lawson, B., 1990, How Designers Think – 2nd Edition, Routledge, London

125- Lawson, B., 1993, The Art of the Process, London: RIBA, pp. 6-10.

126- Lang, Jon, (1987). Creating Architectural Theory: The Role of the Behavioral Sciences in Environmental Design. New York: VanNostrand Reinhold Company.

127- Latour, B. (1987) Science in Action: How to Follow Scientists and Engineers through Society. Harvard University Press, Cambridge.

128- Levin, P., H., 1966, Decision Making in Urban Design, *Building Research Station Note EN51/66*. Herts, UK: Building Research Station, Gartson.

129- Liu, C., Shen, J., Ren, Y., and Zheng, H., 2021, Pipes of AI – Machine Learning Assisted 3D Modeling Design CDRF 2020, Proceedings of the 2020 DigitalFUTURES, pp. 17–26

130- Liu, Y., Fang, C., Yang, Z., Wang, X., Zhou, Z., Deng, Q., and Liang, L., 2022, Exploration on Machine Learning Layout Generation of Chinese Private Garden in Southern Yangtze, (Eds.): CDRF 2021, Proceedings of the 2021 DigitalFUTURES, pp. 35–44.

131- Mahmoodi, A., 2001, The design process in architecture: a pedagogic approach using interactive thinking. PhD thesis, University of Leeds.

132- Malaeb, J., and Ma, W., 2019, Artificial Intelligence in Architecture - General Understanding and Prospective Studies, Shanghai Jiao Tong University.

133- Maldonado, M., 2014, Digital Diagrams as Spatial Design Tools, Session 2 - Advanced Research and Digital Tools

134- Mandow, L., Perez-de-la-Cruz, J., Rodriguez-Gavilan, A., and Ruiz-Montiel, M., 2020, Architectural Planning with Shape Grammars and Reinforcement Learning: Habitability and Energy Efficiency, *Engineering Applications of Artificial Intelligence 96* 103909

135- Martinez, N., 2017, Suggestive Drawing Among Human and Artificial Intelligences, Harvard Graduate School of Design, Cambridge, MA.

136- Mavromatidis, L., Bykalyuk, A., and Lequay, H., 2013, Development of Polynomial Regression Models for Composite Dynamic Envelopes' Thermal Performance Forecasting. *Applied Energy Vol. 104*, pp. 379-391

137- McCormack, J., Dorin, A., and Innocent, T., 2004. Generative design: a paradigm for design research. In: Proceedings of Futureground. Design Research Society, Melbourne.

138- McGilchrist, I. (2009). The master and his emissary: The divided brain and the making of the Western world. Yale University Press.

139- Mildenhall, B., Srinivasan, P., Tancik, M., Barron, J., Ramamoorthi, R., and Ng. R., 2020, NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis https://doi.org/10.48550/arXiv.2003.08934

140- Millan, E., Belmonte, M., Boned, F., Gavilanes, J., Perez-de-la-Cruz, J., and Diaz-Lopez, C., 2022, Using Machine Learning Techniques for Architectural Design Tracking: An Experimental Study of The Design of a Shelter, *Journal of Building Engineering 51* 104223

141- Miller, C., and Nagy, Z., 2018, A Review of Unsupervised Statistical Learning and Visual Analytics Techniques Applied to Performance Analysis of Nonresidential Buildings, Renew. *Sustain. Energy Rev. 81* 1365–1377, doi:10.1016/j.rser.2017.05.124.

142- Millet, K., Buehler, F., Du, G., and Kokkoris, M., 2023. Defending Humankind: Anthropocentric Bias in the Appreciation of AI Art. *Computers in Human Behavior 143* (2023), 9 pages. https://doi.org/10.1016/j.chb. 2023.107707

143- Mitchell, W.J., 1977, Computer-Aided Architectural Design, Van Nostrand Reinhold, New York.

144- Mitchell, W. J., 2005. Placing words: Symbols, space, and the city. MIT Press.

145- Mocanu, E., Nguyen, P., Gibescu, M., and Kling, W., 2016, Deep Learning for Estimating Building Energy Consumption, Sustain. *Energy, Grids Networks 6* 91–99, doi:10.1016/j.segan.2016.02.005.

146- Mohammed., M., Khan, M., and Bashier, E., 2017, Machine Learning Algorithms and Applications., Taylor & Francis Group, LLC., Florida, USA/

147- Moretti, L., 1971, Ricera Mathematica in Architettura e Urbanistica. Moebius IV (1), 30-53

148- Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., and Chen, M., 2022, Point-E: A System for Generating 3D Point Clouds from Complex Prompts https://doi.org/10.48550/arXiv.2212.0875

149- Nicholas, P., and Zwiezyscki, M., 2018, Machine Learning for Architectural Design: Practices and Infrastructure, International Journal of Architectural Computing

150- Nilsson, N. J. (1998). Artificial Intelligence: A New Synthesis. Morgan Kaufmann.

151- Ostertagova, E., 2012, Modelling Using Polynomial Regression, SciVerse ScienceDirect, *Procedia Engineering 48*, p. 500-506

152- Ostertagová, E., 2011, Applied Statistic (in Slovak). Elfa Košice, Slovakia, p. 161, ISBN 978-80-8086-171-1.

153- Oxman, R., 2006, Theory and design in the First Digital Age. Des. Stud. 27, 229-265.

154- Oxman, R., 2017. Thinking difference: Theories and models of parametric design thinking. Design Studies, 52, 4–39

155- Peng, X., Chen, R., Yu, K., Ye, F., and Xue, W., 2017, An Improved Weighted k-Nearest Neighbor Algorithm for Indoor Localization. *Wireless Personal Communications, v. 96*, n. 2, p. 2239-2251, 2017.

156- Peters, T., 2013, Computation works: The building of algorithmic thought. Architectural Design, 83(2), 8–15

157- Ploennings, J., and Berger, M., 2023, AI in Architecture Ploennigs and Berger *AI in Civil Engineering (2023)* 2:8 https://doi.org/10.1007/s43503-023-00018-y

158- Poole, D., Mackworth, A., & Goebel, R. (1998). Computational Intelligence: A Logical Approach. Oxford University Press.

159- Popper, K., (1963). Conjectures and Refutations: The Growth of Scientific Knowledge. London: Routledge and Kegan Paul.

160- Qin, W., Chen, L., Zhang, B., Chen, W., and Luo, H., 2023, NeoDescriber: An Image-To-Text Model for Automatic Style Description of Neoclassical Architecture, *Expert Systems with Applications 231* (2023) 120706

161- Quinlan, J., 1993, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, Massachusetts, USA.

162- Ragot, M., Martin, N., and Cojean, S., 2020. AI-Generated vs. Human Artworks. A Perception Bias Towards Artificial Intelligence? In Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHIEA '20). Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3334480.3382892

163- Rahman, A., 2018, Applications of Machine Learning, Deep Learning and Reduced-Order Models, Mechanincal Engineering department, The University of Utah.

164- Rahman, A. and Smith, A., 2018, Predicting Heating Demand and Sizing a Stratified Thermal Storage Tank Using Deep Learning Algorithms, *Appl. Energy. 228*, 108–121, doi:10.1016/j.apenergy 2018.06.064.

165- Rahman, A. and Smith, A., 2017, Predicting Fuel Consumption For Commercial Buildings With Machine Learning

Algorithms, *Energy Build 152* 341–358, doi:10.1016/j.enbuild.2017.07.017.

166- Rahman, A., Srikumar, V., Smith, A., 2018, Predicting Electricity Consumption For Commercial And Residential Buildings Using Deep Recurrent Neural Networks, *Appl. Energy 212* 372–385, doi:10.1016/j.apenergy.2017.12.051

167- Reffat, R., 2008, Investigating Patterns of Contemporary Architecture Using Data Mining Techniques, *Digital Applications in Construction, eCAADe 26*, pp. 601-608.

168- Rich, E., & Knight, K. (1991). Artificial Intelligence (2nd ed.). McGraw-Hill.

169- Rittel, H. W. J., 1972, On the Planning Crisis: Systems Analysis of the first And Second Generation. *Bed-riftsøkonomen, 8*: 390–398; Translated in Protzen and Harris (2010), pp. 151–165.

170- Robert, J. K., 2002, Architectural Interpretation of Cellular Automata. Generative Art.

171- Robinson, C., et., Al, 2017, Machine Learning Approaches For Estimating Commercial Building Energy Consumption, Appl. Energy 208 889–904, doi:10.1016/j.apenergy.2017.09.060.

172- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B., 2022. High-resolution image synthesis with latent diffusion models. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10684–10695.

173- Rowe, P., 1987, Design Thinking, The Massachusetts Institute of Technology.

174- Roy, W., Sari, M., Berawi, M., Zagloel, T., Madyaningarum, N., Miraj, P., Pranoto, A., and Susantono, B., 2023, Machine Learning-Based Energy Use Prediction for The Smart Building Energy Management System

175- Royal College of Art (1979). Design in general education. London, UK: Department of Design Research, Royal College of Art.

176- Russell, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.

177-Rzevski, G., Woolman, D., and Trafford, D.B., (1980). Validation of Design Methodology, *Design Studies, Vol. 1, Issue 6*, pp. 325-328

178-Salvadori, Mario G., (1974). "Is Automated Design Possible?" In: R. Spillers ed. Basic Questions of Design Theory. New York: American Elsevier. pp. 95-102.

179-Samuel, A., 1959, Some Studies in Machine Learning Using the Game Of Checkers. *IBM Journal of Research and Development, 3(3)*, 210-229.

180-Schalkoff, R.J., 1997. Artificial Neural Networks. McGraw-Hill, New York.

181-Schon, D., (1984). The Reflective Practitioner. New York: Basic Books.

182-Schumacher, P., 2008, Parametricism as Style: Parametric Manifesto. In: *The Darkside Club, 11th Architecture Biennale, Venice*, 11 Septembler.

183-Sternberg, R. J. (1997). Thinking styles. New York: Cambridge University Press.

184- Simon, Herbert A., (1957). Models of Man. New York: John Wiley.

185- Simon, Herbert A., (1960). The New Science of Management Decisions. N.Y.: Harper.

186- Simon, Herbert, (1969). The Sciences of the Artificial. Cambridge, Mass.: MIT Press.

187-Khean, N., Fabbri, A., & Haesler, MH 2018 'Learning Machine Learning as an Architect, How To? Presenting And Evaluating a Grasshopper Based Platform to Teach Architecture Students Machine Learning', *Computing A Better Tomorrow - 36th eCAADE*, Lodz, Poland, pp. 95-102

188-Sebestyen, A., 2020, Machine Learning Methods in Energy Simulations for Architects and Designers - The Implementation of Supervised Machine Learning in The Context of The Computational Design Process, *Architecture and Fabrication in the cognitive age, eCAADe 2020.*

189-Segal, M., 2003, Machine Learning Benchmarks and Random Forest Regression, Center for Bioinformatics & Molecular Biostatistics, University of California, San Francisco

190-Sekhar, S., Roy, R., and Balas, V., 2018, Estimating Heating Load In Buildings Using Multivariate Adaptive Regression Splines, Extreme Learning Machine, A Hybrid Model of MARS and ELM, Renew. Sustain. *Energy Rev. 82* 4256–4268, doi:10.1016/j.rser.2017.05.249.

*191*-Shakiban C. and Berstedt J.E., 1998, Generalized Koch Snowflakes. In *Bridges: Mathematical Connections in Art, Music, and Science*

192-Shalunts, et. Al., 2011, Architectural Style Classification of Building Façade Windows, Advances in Visual Computing - 7th International Symposium, ISVC 2011, Las Vegas, NV, USA, September 26-28, 2011. Proceedings, Part II

193-Singh, M., Haxhimusa, Y., and Sablatnig, R., 2022, Early-Stage Design Support Combining Machine Learning and Building Information Modelling, *Automation in Construction – 136*

194-Singh, V., and Gu, N., (2012). Towards an Integrated Generative Design Framework, *Design Studies, Vol. 33, Issue 2*, pp. 185-207.

195- Soleimani, A., 2019, Computational Design Thinking and Thinking Design Computing, University of Oregon | College of Design School of Architecture & Environment

196- Sidey-Gibbons, J., & Sidey-Gibbons, C., 2019, Machine Learning in Medicine: A Practical Introduction. *BMC Medical Research Methodology, 19*, http://dx.doi.org/10.1186/s12874-019-0681-4.

197- Soofi, A., 2017, Classification Techniques in Machine Learning: Applications and Issues, J*ournal of Basic & Applied Sciences, Vol. 13*, pp. 459-465

198- Soleimani, A., 2019, Computational Design Thinking and Thinking Design Computing, *Reynolds Symposium: Education by Design. Portland, Oregon*, October 18-19, 2019.

199- Stiny, G., 1985, Computing with Form and Meaning in Architecture. *Journal of Architectural Education, 39(1)*: 7-19

200- Stiny, G., and Gips, J., Shape Grammars and the Generative Specification of Painting and Sculpture, C V Freiman (ed.) *Information Processing 71* (Amsterdam: North-Holland, 1972) 1460-1465.

201- Studer, Raymond, (1970). "The Dynamics of Behaviour-Contingent Physical Systems". In: Harold Proshansky et al. Ed.

Environmental Psychology: Man and his Physical Setting. New York: Holt, Reinehart and Winston.

202- Sun, C., Zhou, Y., and Han, Y., 2022, Automatic Generation of Architecture Facade for Historical Urban Renovation Using Generative Adversarial Network, *Building and Environment 212* (2022) 108781

203- Sun, M., Zhang, F., Duarte, F., and Ratti, C., 2022, Understanding Architecture Age and Style Through Deep Learning, *Cities 128* 103787

204- Szeliski, R. (2010). Computer vision: Algorithms and applications. Springer.

205- Tamke, M., Nicholas, P., and Zwierzycki, M., 2018, Machine Learning for Architectural Design: Practices and Infrastructure, *International Journal of Architectural Computing 16(2)*: pp. 123-143

206- Takizawa, A., and Furuta, A., 2017, 3D Spatial Analysis Method with First-Person Viewpoint by Deep Convolutional Neural Network with Omnidirectional RGB and Depth Images, Proceedings of the 35th eCAADe Conference At: Sapienza University of Rome, Italy Volume: 2

207- Tansas, A., and Xifara, A., Accurate Quantitative Estimation of Energy Performance of Residential Buildings Using Statistical Machine Learning Tools, *Energy Build 49* (2012) 560–567 https://doi.org/10.1016/j.enbuild.2012.03.003

208- Tedeschi, A., 2014, AAD Algorithms-Aided Design: Parametric Strategies Using Grasshopper, Le Penseur, Italy.

209- Terzidis, K., 2003. Expressive Form: A Conceptual Approach to Computational Design. Spon Press, New York.

210- Terzidis, K., 2004. Algorithmic design: a paradigm shift in Architecture?. In: Proceedings of the 22nd eCAADe Conference, pp. 201e207.

211- Terzidis, K., 2006, Algorithmic Architecture, first ed. Elsevier Ltd.

212- Tovey, M., 1984, Designing with Both Halves of the Brain, *Design Studies 5*, pp. 219-228

213- Topuz, B., & Alp, N. (2023). Machine Learning in Architecture. *Automation in Construction, Vol 154*.

214- Turing, A. M. (1950). Computing Machinery And Intelligence. Mind, 59(236), 433–460.

215- Uyanik, G., & Guler, N., 2013, A Study on Multiple Linear Regression Analysis., *Procedia - Social and Behavioral Sciences 106*, pp. 234-240.

216- Uzair, M., Chun, D., Zeeshan, Han, H., Jeon, G., and Chen, K., 2019, A Review of The Applications of Artificial Intelligence and Big Data to Buildings for Energy-Efficiency and A Comfortable Indoor Living Environment. *Energy & Buildings – Science Direct*

217- Uzun, C., and Colakoglu, M., 2019, Architectural Drawing Recognition A Case Study for Training the Learning Algorithm with Architectural Plan and Section Drawing Images, Conference: 37 Education and Research in Computer Aided Architectural Design in Europe and XXIII Iberoamerican Society of Digital Graphics, Joint Conference (N. 1).

218- Wallas, G. (1926). The art of thought. New York, NY: Harcourt, Brace and Company.

219- Wang, Bo., Zhang, S., Zhang, J., and Cai, Z., 2022, Architectural Style Classification Based on CNN And Channel–Spatial Attention, Signal, Image, and Video Processing *Springer Nature 2022*

220- Witt, A., 2018, Grayboxing. Log, 43, 69–77. https://www.jstor.org/stable/26588482

221- Wu, L., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J., 2016, Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling, NeurIPS

*222-*Xu, Z., Tao, D., Zhang, Y., Wu, J., and Tsoi, A., 2014, - Architectural Style Classification Using Multinomial Latent Logistic Regression, (Eds.): ECCV 2014, Part I, LNCS 8689, pp. 600–615, 2014. *Springer International Publishing Switzerland 2014*

223-Yang, J., Rivard, H., and Zmeureanu, R., 2005, On-Line Building Energy Prediction Using Adaptive Artificial Neural Networks, *Energy Build 37* (2005) 1250–1259, doi:10.1016/j.enbuild.2005.02.005.

224-Yazici, S., 2020, A Machine Learning Model Driven by Geometry, Material, and Structural Performance Data in Architectural Design Process, *Making Through Code – Built by Data and The Architectural Illustration Vol.1, eCAADe 38*, pp. 411-418.

225-Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *Proceedings of the 33rd Annual*

*Meeting on Association for Computational Linguistics* (pp. 189-196).

226- Yun, L., Luck, R., Mago, P., and Cho, H., 2012, Building Hourly Thermal Load Prediction Using An Indexed ARX Model, *Energy Build 54* (2012) 225–233, doi:10.1016/j.enbuild.2012.08.007.

227- Yieldiz, B., Bilbao, J., and Sproul, A., 2017, A Review and Analysis of Regression and Machine Learning Models On Commercial Building Electricity Load Forecasting, Renew. Sustain. *Energy Rev. 73* 1104–1122, doi:10.1016/j.rser.2017.02.023

228- Yilmaz, S., 1999, Evolution of the Architectural Form Based on The Geometrical Concepts, PhD Thesis (Izmir: Institute of Technology).

229- Yilmaz, G., and Koparan, T., 2016, The Effect of Designed Geometry Teaching Lesson to the Candidate Teachers' Van Hiele Geometric Thinking Level, Journal of Education and Training Studies.

230- Yu, Z., Fung, B. C., Haghighat, F., Yoshino, H., & Morofsky, E. (2010). A Decision Tree Method for Building Energy Demand Modeling. *Energy and Buildings, 42(10)*, 1637–1646.

231- Yucong, W., & Bo, W. (2020). EA-XGBoost: A Novel Evolutionary Algorithm Based XGBOOST Model for Predicting Energy Consumption in Buildings. *Energy Reports, 6*, 2189–2197.

232- Zarei, Y., 2012, The Challenges of Parametric Design in Architecture Today: Mapping the Design Practice. Master Thesis. University of Manchester.

233-.Zboinska, M.A., 2015, Hybrid CAD/E Platform Supporting Exploratory Architectural Design. *CAD Computer Aided Design 59*, 64-84.

234-Zhang, P., & Wei-guo, X., 2018. Quasicrystal Structure Inspired Spatial Tessellation in Generative Design.

235-Zheng, H., and Yuan, P., 2021, A Generative Architectural and Urban Design Method Through Artificial, *Building and Environment 205* (2021) 108178

236-Zheng, H., (2018), Architectural Drawings Recognition and Generation through Machine Learning, *ACADIA 2018*.

## *Websites:*

1- Bevans, R., 2020, "Simple Linear Regression: An Easy Introduction & Examples", https://www.scribbr.com/statistics/simple-linear-regression/, Last Access: 18-8-2023.

2- Biswal, A. (2023). 7 Types of Artificial Intelligence That You Should Know in 2023. Simplilearn. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/types-of-artificial-intelligence, Last Access: 24-9-2024.

3- Caldwell, B., Dake, D., Sadly, M., and Ulch, L., 2000, Integrated HOTS Thinking Model, Department of Art and Design, Iowa State University.

4- Great learning Team, 2022, What is Ridge Regression, https://www.mygreatlearning.com/blog/what-is-ridge-regression/, Last Access: 16-8-2023.

5- Hohenadel, K., (2020). What Is Contemporary Architecture? https://www.thespruce.com/what-is-contemporary-architecture-4769079, Last visit: 21/4/2021.

6- Kriesel, D., 2005, A brief introduction to Neural networks, Univ. Bonn Semin. Proc. http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf.
7- Malik, S, Harode, R., and Kunwar, A., 2020, XGBoost: A Deep Dive into Boosting, https://medium.com/sfu-cspmp/xgboost-a-deep-dive-into-boosting-f06c9c41349 Last Accessed: 16-8-2023.
8- McGee, V., 2023, What Is Coding and What Is It Used For? https://www.computerscience.org/resources/what-is-coding-used-for/, Last Accessed: 24-August, 2023
9- Rayman, P., 2017, What Is Transformation Matrix and How to Use It, https://forum.patagames.com/posts/t501-What-Is-Transformation-Matrix-and-How-to-Use-It, Last Accessed: 24-August, 2023
10- Sawantt, S., 2021, Grasshopper 3D: A Modeling Software Redefining The Design Process, https://parametric-architecture.com/grasshopper-3d-a-modeling-software-redefining-the-design-process/, last access: 14/10/2024.
11- Solomonoff, G., 2023, The Meeting of the Minds That Launched AI, https://spectrum.ieee.org/dartmouth-ai-workshop, last access: 14/10/2024.

# Appendices

## *Appendix A (Glossary)*

***Accuracy:*** A metric in classification that measures the ratio of correctly predicted instances to the total instances.

***AdaBoost (Adaptive Boosting):*** An ensemble learning method that combines weak learners into a strong learner. It assigns weights to instances in the dataset, allowing subsequent weak learners to focus on the misclassified instances from the previous ones.

***Algorithm:*** A finite sequence of instructions followed by a computer system.

***Algorithmic Design:*** Design process that relies on algorithms, step-by-step procedures, or formulas, to generate and manipulate design elements.

***API (Application Programming Interface):*** A set of rules and tools that allows different software applications to communicate with each other. APIs define the methods and data formats for requests and responses.

***Artificial intelligence (AI):*** The simulation of human intelligence in machines that are programmed to think and learn, enabling them to perform tasks that typically require human intelligence.

***Artificial general intelligence (AGI):*** The representation of generalized human cognitive abilities in software so that, faced with an unfamiliar task, the AGI system could find a solution.

***Artificial narrow intelligence (ANI):*** A type of AI that is focused on performing a specific task or set of tasks.

***Artificial super intelligence (ASI):*** A speculative type of AI that surpasses human intelligence in all respects.

***Artificial Neural Network (ANN):*** Computational models inspired by the structure and function of biological neural networks, used in machine learning to recognize patterns and make decisions.

***Area Under the Curve (AUC):*** The area under the Receiver Operating Characteristic (ROC) curve. AUC is a single value summary of the ROC curve, where a higher AUC generally indicates better model performance.

***Automation:*** Handling a process with machines or software so that less human input is needed.

***Bagging*** An ensemble learning technique that leverages bootstrapping to improve the performance and robustness of machine learning models. In bagging, multiple models (often of the same type, such as decision trees) are trained on different bootstrap samples of the dataset. The final prediction is obtained by aggregating the predictions of individual models, typically through averaging (for regression) or voting (for classification). Bagging helps reduce overfitting and enhances the model's generalization capability.

***Bard:*** A chatbot developed by Google, released in March 2023.

***Best-Fit Line:*** In statistics, the line that best represents the relationship between two variables, typically determined through methods like linear regression.

***Bias:*** The assumptions that an AI makes to simplify its tasks.

***Big data:*** Very large datasets that normal data-processing software can’t handle.

***Bing Chat:*** An AI chatbot feature integrated into Bing, released in February 2023.

***Boosting***: An ensemble learning technique where multiple weak models are combined sequentially to create a strong model, with each model giving more weight to misclassified instances.

***Bootstrapping***: a resampling technique in which multiple subsets, called bootstrap samples, are generated by randomly sampling with replacement from the original dataset. It is commonly used in statistics and machine learning to assess the variability of a statistic, estimate confidence intervals, or improve the robustness of model training.

***Categorical Data***: Data which consists of variables that can take on a limited set of discrete values or categories.

***Chatbot***: A software application that mimics human conversation, usually through text.

***ChatGPT***: An AI chatbot released by OpenAI in November 2022

***Classification:*** A type of supervised machine learning task where the goal is to categorize input data into predefined classes or categories.

***Classification Report:*** A summary of the performance of a classification model. It typically includes metrics such as precision, recall, F1 score, and accuracy, providing a comprehensive evaluation of how well the model is classifying instances into different classes.

***Coding:*** The process of writing instructions for a computer to execute, typically using a programming language.

***Computational Design:*** The use of computational tools and techniques, often involving algorithms, to assist in the creation and manipulation of design elements.

***Computer Vision:*** A field of artificial intelligence that focuses on enabling machines to interpret and understand visual information from the world, often involving image and video analysis.

***Continuous Data***: Data which represents variables that can take an infinite number of values within a given range.

***CNN (Convolutional Neural Network):*** A type of neural network architecture designed for processing structured grid data, particularly images. It uses convolutional layers to automatically learn hierarchical features from the input data.

***Correlation Matrix***: A table showing correlation coefficients between variables. Each cell in the table represents the correlation between two variables, with values ranging from -1 to 1. It is often used in statistics and data analysis to understand relationships between different variables.

***Cost Function***: A function that measures the difference between predicted values and actual values, guiding the learning process in machine learning algorithms.

***Cross-Validation:*** A technique used to assess the performance of a machine learning model by dividing the dataset into multiple subsets. The model is trained on some of these subsets and tested on the remaining subset. This process is repeated multiple times, and performance metrics are averaged, providing a more robust evaluation of the model's generalization capability.

***Data Resampling***: Techniques such as oversampling or undersampling used to address imbalances in class distribution within a dataset.

***Data set in ML and AI***: A collection of data used for training, testing, and validating machine learning and artificial intelligence models. It typically includes input features and corresponding output labels for supervised learning or only input features for unsupervised learning.

***Decision Trees:*** A fundamental machine learning algorithm that recursively splits the data based on features to make decisions. Each internal node represents a decision based on a feature, and each leaf node represents an output or class label.

***Deep learning***: A subfield of machine learning that involves the use of artificial neural networks, particularly deep neural networks with multiple layers (deep architectures). These networks are capable of automatically learning hierarchical representations of data, leading to powerful models for tasks such as image recognition, natural language processing, and more.

***Diffusion***: The process by which something (e.g., information, substances) spreads or moves from one place to another within a medium. In the context of machine learning and AI diffusion models are a class of generative models that use a diffusion process to model the generation of images. In these models, a latent image is iteratively transformed through a series of steps, introducing noise at each step. The process gradually refines the image, and the noise is controlled in a way that enables the generation of high-quality and diverse images.

***Discriminator***: A neural network that evaluates input data and tries to distinguish between real and generated data. The goal of the discriminator is to correctly classify whether the input data comes from the real dataset or was produced by the generator.

***Dimensionality Reduction:*** The process of reducing the number of features (variables) in a dataset. It aims to retain the most important information while minimizing the loss of data, often improving computational efficiency, and mitigating the curse of dimensionality.

***Ensemble Learning:*** A machine learning technique where multiple models are combined to improve overall performance and accuracy. Common methods include bagging (e.g., Random Forest) and boosting (e.g., AdaBoost).

***Exploratory Data Analysis (EDA):*** The process of visually and statistically analyzing datasets to uncover patterns, trends, and anomalies before applying machine learning algorithms.

***F1 Score:*** A metric in classification that combines precision and recall into a single value, balancing false positives and false negatives.

***Feature importance***: The measure of the impact of each feature (input variable) on the model's predictions.

***Features:*** The input variables or attributes used by machine learning algorithms to make predictions.

***Form Finding***: In design and engineering, it refers to the process of determining the optimal form or shape of a structure based on specified constraints and criteria.

***Form Making:*** In the context of design, it generally refers to the process of creating physical or digital forms, shapes, or structures.

***GAN (Generative Adversarial Network):*** A type of artificial intelligence model consisting of two neural networks, a generator, and a discriminator, trained adversarially to generate realistic data.

***Generative AI***: AI systems that generate output according to a learning mechanism.

***Generative Design:*** An approach in design and engineering where algorithms are used to explore a range of possible design solutions based on specified criteria, enabling the creation of innovative and optimized designs.

***Generative pre-trained transformer (GPT):*** A type of LLM used in ChatGPT and other AI applications.

***Generator***: A neural network responsible for generating synthetic data. It takes random noise as input and transforms it into data that ideally is indistinguishable from real data.

***Gradient Descent:*** An optimization algorithm used to minimize the cost function in machine learning by adjusting model parameters iteratively in the direction of steepest descent.

***K-Fold Cross-Validation***: A specific type of cross-validation where the dataset is divided into 'k' subsets or folds. The model is trained 'k' times, each time using a different fold as the test set and the remaining folds as the training set. The performance metrics are then averaged over the 'k' iterations, providing a more reliable estimate of the model's performance compared to a single train-test split. Common values for 'k' are 5 or 10 in practice.

***k-NN (k-Nearest Neighbors):*** A type of algorithm used for classification and regression tasks, where an object is classified by the majority vote of its k nearest neighbors.

***Large language model (LLM):*** A neural net trained on large amounts of text to imitate human language.

***Machine learning (ML):*** The study of how AI acquires knowledge from training data.

***Mean Absolute Error (MAE):*** A measure of the average absolute difference between predicted and actual values in a regression problem.

***Mean Squared Error (MSE):*** A metric in regression analysis that measures the average squared difference between predicted and actual values.

***MidJourney:*** An AI image generator released in July 2022.

***Natural language processing (NLP):*** The study of interaction between computers and human language.

***NeRF (Neural Radiance Fields):*** a novel approach to 3D scene representation and rendering using neural networks. It models a scene as a continuous 3D function that maps 3D spatial coordinates to scene color and density. NeRF has been applied to generate highly detailed and realistic renderings of scenes, making it particularly useful for computer graphics and virtual reality applications.

***Non-Generative AI***: AI models or systems that don't generate new content or data but instead focus on analyzing, classifying, or making predictions on existing data.

***Normalization:*** The scaling of features to a standard range, often between 0 and 1. It ensures that different features with varying scales contribute equally to the model, preventing dominance by features with larger magnitudes.

***OpenAI:*** A leading AI company that developed ChatGPT and DALL-E.

***One-Hot Encoding:*** is a technique to represent categorical variables as binary vectors. Each category is mapped to a unique binary value, creating a sparse matrix where only one element is "hot" (1) while others are "cold" (0).

***Overfitting:*** A modeling error that occurs when a machine learning algorithm captures noise or random fluctuations in the training data, leading to poor performance on new, unseen data.

***Parameter:*** A variable in an AI system that it uses to make predictions.

***Parametric Design:*** A design approach that uses parameters and rules to create variations within a system, allowing for flexible and dynamic designs based on changing parameters.

***Precision***: A metric in classification that measures the proportion of predicted positive instances that are actually positive.

***Principal Component Analysis (PCA):*** a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional representation while retaining as much of the original variance as possible. It identifies the principal components, which are linear combinations of the original features, and ranks them by their ability to explain variance.

***Programming:*** The process of giving instructions to a computer (using computer code).

***Prompt:*** The input from the user to which the AI system responds.

***$R^2$ Score (Coefficient of Determination):*** A metric that represents the proportion of the variance in the dependent variable that is predictable from the independent variables.

***Random Forest:*** An ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes for classification tasks or the average prediction for regression tasks.

***Recall (Sensitivity or True Positive Rate):*** A metric in classification that measures the proportion of actual positive instances correctly predicted by the model.

***Regularization:*** A technique used in machine learning to prevent overfitting by adding a penalty term to the model's cost function. It discourages overly complex models by penalizing large coefficients, promoting a balance between model complexity and accuracy.

***Reinforcement Learning:*** A type of machine learning where an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on the actions it takes.

***Regression:*** A statistical technique used to model and analyze the relationship between a dependent variable and one or more independent variables. Or a type of supervised learning where the goal is to predict a continuous outcome variable based on one or more input features.

***RhinoCommon***: A .NET-based framework for developing software that integrates with Rhino, a 3D modeling software. It allows developers to create custom applications, plugins, and scripts to automate design and modeling tasks within Rhino.

***Ridge:*** Refers to Ridge Regression, a linear regression technique that adds a penalty term based on the squared values of the coefficients, helping to prevent overfitting.

***RNN (Recurrent Neural Network):*** A type of neural network architecture that is well-suited for processing sequential data. It has connections that allow information to be passed from one step of the sequence to the next, enabling it to capture temporal dependencies in data.

***ROC Curve (Receiver Operating Characteristic)***: A graphical representation of the trade-off between the true positive rate and false positive rate at various thresholds usually used with classification problems.

***Root Mean Squared Error (RMSE):*** The square root of the mean squared error, providing a measure of the average magnitude of errors in predictions.

***Scikit-Learn***: An open-source machine learning library for Python. It provides a variety of tools for data analysis and machine learning, including algorithms for classification, regression, clustering, and model evaluation.

***Scripting:*** Writing scripts, which are sets of instructions written in a scripting language, for automating tasks or processes.

***SDK (Software Development Kit):*** A set of software development tools that allows developers to create applications for a certain software package, hardware platform, computer system, or operating system.

***SMOTE (Synthetic Minority Over-sampling Technique):*** A technique used in machine learning to address class imbalance by generating synthetic samples for the minority class.

***Stable Diffusion:*** A process where the spreading or movement of a substance or information is steady and does not result in rapid or extreme changes. Stability in diffusion implies a more controlled and gradual progression.

***Supervised Learning***: A type of machine learning where the model is trained on a labeled dataset, meaning it is provided with input-output pairs to learn the mapping between input data and corresponding output.

***Support Vector Machine (SVM):*** A supervised machine learning algorithm used for classification and regression tasks. SVM finds the hyperplane that best separates data into different classes in a high-dimensional space. It aims to maximize the margin between classes and can handle linear and non-linear relationships through the use of different kernel functions.

***Targets***: The output variable that machine learning algorithms aim to predict.

***Test Data Set***: A subset of data used to evaluate the performance of a trained machine learning model on new, unseen instances.

***Training Data Set***: The subset of data used to train a machine learning model.

***Turing test:*** A test of a machine's ability to display human intelligence.

***Underfitting:*** A modeling error that occurs when a machine learning algorithm is too simple to capture the underlying patterns in the training data, resulting in poor performance on both the training and new data.

***Unsupervised Learning***: A type of machine learning where the model is trained on unlabeled data, and the system tries to learn the patterns and relationships within the data without explicit guidance on the output.

***Validation Data Set:*** A separate subset of data used to tune and optimize model hyperparameters during training.

***XGBoost (Extreme Gradient Boosting):*** An efficient and scalable implementation of gradient boosting. It is designed for speed and performance and is widely used for both classification and regression tasks.

# *Appendix B (Formulas and Equations)*

1- Regression

***Linear regression:***

Any linear regression equation (without error) takes the following form:

$$\hat{Y} = bX + a$$

Where:
$\hat{Y}$: Predicted values of Y

b: Slope = Rate of predicted ↑/↓ for Y scores for each unit increase in X.

a: Y-intercept = level of Y when X = 0

***Univariate linear regression:***

Univariate linear regression focuses on determining the relationship between one independent (explanatory variable) variable and one dependent variable.

In a linear regression equation, the hypothesis, parameters, cost function, and goals are determined as follows:

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: $\theta_0, \theta_1$

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

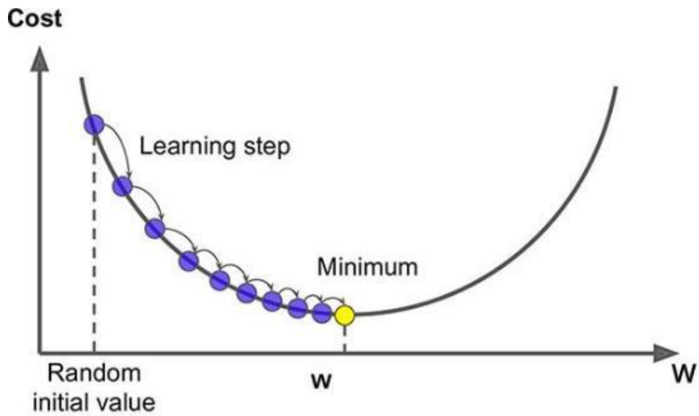
Goal: $minimize\, J(\theta_0, \theta_1)$

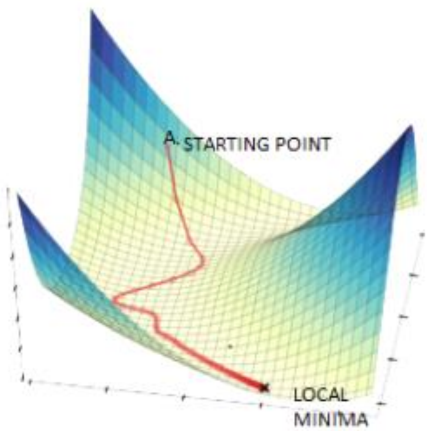
In order to minimize the cost, a gradient descent method is used which has the following equation:

Repeat until convergence: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

Where $\alpha$ is the learning rate of the algorithm which can control the jump size of the weight update in each iteration.

The gradient descent graph in both 2D and 3D spaces could be presented as follows:




*Gradient Descent in 2D Space (Left) and in 3D Space (Right)*
*(https://www.analyticsvidhya.com/blog/2020/10/what-does-gradient-descent-actually-mean/, https://www.hackerearth.com/blog/developers/3-types-gradient-descent-algorithms-small-large-data-sets/)*

***General Gradient Descent Equations:***

$$\theta_0 := \theta_0 - \alpha \, \frac{1}{m} \textstyle\sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \, \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) \, x_i)$$

***Linear regression with Multi Variables (Multivariate)***

Multivariate regression is a technique that estimates a single regression model with more than one outcome variable.

n = number of features

$x^{(i)}$ = input (features) of $i^{th}$ training example

$x_j^{(i)}$ = value of features j in $i^{th}$ training example (j: row number, i: column number)

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ … Multivariate

For convenience of notation, define $x_0 = 1$.

$h_\theta(x) = [\theta_0 \ \ \theta_1 \ \ \theta_2 \ \ldots \ \theta_n] \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \theta^T \cdot x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Note that X = $\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \epsilon \, \mathbb{R}^{n+1}$ and $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \cdots \\ \theta_n \end{bmatrix} \epsilon \, \mathbb{R}^{n+1}$ and $\theta^T = [\theta_0 \ \ \theta_1 \ \ \theta_2 \ \ldots \ \theta_n]$ = (n + 1) x 1 matrix..

Parameters: $\theta_0, \theta_1, \ldots \theta_n$

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

Goal: $minimize\ J\left(\theta_0, \theta_1, \theta_2, \ldots \theta_n\right)$

In this scenario, the gradient descent equation will be:

Repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) . x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) . x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) . x_2^{(i)}$$

…

}

And the general gradient descent equation will be:

Repeat until convergence: {

$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right).x_j^{(i)}$ for j := 0…n}

***XGBoost***

XGBoost is one of the most popular and efficient implementations of the Gradient Boosted Trees algorithm, a supervised learning method that is based on function approximation by optimizing specific loss functions as well as applying several regularization techniques.

The objective function (loss function and regularization) at iteration t that needs to be minimized is the following:

The gradient boosting ensemble technique operates through a series of straightforward steps to enhance the predictive power of a model:

*Initialization:* A base model ($F_0$) is established to make initial predictions for the target variable y. The differences between the actual values and ($F_0$) predictions represent the residuals.

*Model Fit to Residuals*: A new model ($h_1$) is then fitted specifically to the residuals obtained from the initial model. This second model is designed to capture the patterns or information that the first model failed to predict accurately.

*Boosting and Model Combination*: The boosted model ($F_1$) is formed by combining the initial model ($F_0$) with the new model ($h_1$) This amalgamation results in an improved model ($F_1$) with a lower mean squared error compared to ($F_0$). The process can be iteratively repeated for (m) iterations, with each new model capturing and refining the residuals from the previous ensemble, ultimately reducing prediction errors to the extent possible. This iterative approach enhances the overall performance of the ensemble by addressing deficiencies in the predictions made by earlier models.

$$F_m(x) <- F_{m-1}(x) + h_m(x)$$

F0(x) should be a function which minimizes the loss function or MSE (mean squared error), in this case:

$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_{i^2} \gamma)$$

$$argmin_\gamma \sum_{i=1}^{n} L(y_{i^2} \gamma) = argmin_\gamma \sum_{i=1}^{n} (y_i - \gamma)^2$$

Taking the first differential of the above equation with respect to γ, it is seen that the function minimizes at the mean i=1nyin. So, the boosting model could be initiated with:

$$F_0(x) = \frac{\sum_{i=1}^{n} y_i}{n}$$

$F_0(x)$ gives the predictions from the first stage of the model. Now, the residual error for each instance is $(y_i - F_0(x))$.

The residuals from $F_0(x)$ could be used to create $h_1(x)$. $h_1(x)$ will be a regression tree which will try and reduce the residuals from the previous step. The output of $h_1(x)$ won't be a prediction of y; instead, it will help in predicting the successive function $F_1(x)$ which will bring down the residuals.

The additive model $h_1(x)$ computes the mean of the residuals $(y - F_0)$ at each leaf of the tree. The boosted function $F_1(x)$ is obtained by summing $F_0(x)$ and $h_1(x)$. This way h1(x) learns from the residuals of $F_0(x)$ and suppresses it in $F_1(x)$.

This can be repeated for 2 more iterations to compute $h_2(x)$ and $h_3(x)$. Each of these additive learners, $h_m(x)$, will make use of the residuals from the preceding function, $F_{m-1}(x)$.

2- Classification

The specific equations used in classification in machine learning can vary depending on the algorithm being employed.

***<u>Logistic Regression:</u>***

The logistic regression equation models the probability that the dependent variable (x) is 1 as a function of the independent variables ($x_1$, $x_2$, …, $x_n$) and their corresponding coefficients ($b_1$, $b_2$, …, $b_n$).

Logistic regression uses a logistic function called a sigmoid function to map predictions and their probabilities. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1.

The sigmoid function $(\frac{1}{1+e^{-z}})$ ensures that the output is between 0 and 1.

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

Where:

f(x) is the output of the function.

L is the curve's maximum value.

e is bae of natural logarithms

K is logistic growth rate or steepness of the curve.

$x_0$ is the x value of the sigmoid midpoint.

and x is a real number.

The functions could be written as:

$$P = \frac{1}{1 + e^{-(a+bX)}}$$

where P is the probability of a 1 (the proportion of 1s, the mean of Y), e is the base of the natural logarithm (about 2.718) and a and b are the parameters of the model. The value of a yields P when X is zero, and b adjusts how quickly the probability changes with changing X a single unit (There can be standardized and unstandardized b weights in logistic regression, just as in ordinary linear regression). Because the relation between X and P is nonlinear, b does not have a straightforward interpretation in this model as it does in ordinary linear regression.

***Decision Trees***

Decision trees employ a tree-like flowchart structure to illustrate predictions derived from a sequence of feature-based divisions. The process initiates at a root node, where the dataset is split based on specific features. Subsequently, this branching continues until reaching terminal nodes known as leaves, where final decisions or predictions are made based on the characteristics of the data within those leaves.

In the context of decision trees, the concept of impurity is crucial in determining how to split the data effectively. The goal is to create splits that lead to homogeneous subsets, where all instances share the same class label, making the split "pure." A "pure" split means that after the split, the resulting subsets ideally contain instances belonging to only one class, making it easier to make accurate predictions for that subset. In binary classification, this would mean that a split results in subsets where one contains instances labeled "yes" and the other contains instances labeled "no." The decision tree algorithm aims to iteratively create such pure splits to effectively classify instances based on the selected features. The measure of impurity (or purity) helps guide this decision. Common impurity measures include Gini impurity and entropy. Mathematically Gini index can be written as:

$$Gini\ Index = 1 - \sum_{i=1}^{n} (P_i)^2$$
$$= 1 - [(P_+)^2 + (P_-)^2]$$

Where $P_+$ is the probability of a positive class and P_ is the probability of a negative class.

For the right split, the Gini Index will be 0.5.

Weighted Gini index is calculated afterwards. That is the total Gini index of this split. Similarly, this algorithm will try to find the Gini index of all the splits possible and will choose that feature for the root node which will give the lowest Gini index. The lowest Gini index means low impurity.

Entropy has a mathematical formula using logarithmic function as follows:

$$E(S) = -p_{(+)}logp_{(+)} - p_{(-)}logp_{(-)}$$

However, many boosting algorithms use the Gini index as their parameter because logarithmic calculations present in the entropy equation take more time than the Gini index.

The key distinction between decision trees and random forests lies in their approach to modeling and predicting outcomes. Random Forest, classified as a bagging method, deviates from the singular nature of decision trees by constructing an ensemble of decision trees. This ensemble is created by training each tree on a distinct subset of the original dataset, a process known as bootstrapping. This unique feature significantly contributes to mitigating overfitting, a common challenge associated with individual decision trees that tend to capture noise in the training data. By aggregating the predictions of multiple trees, Random Forest produces a more robust and generalized model. Furthermore, its versatility is evident in its applicability to both classification and regression problems. In classification tasks, the ensemble combines the class predictions of individual trees, while in regression tasks, it averages the predictions for continuous outcomes. The utilization of Random Forest thus stands as an effective strategy for enhancing predictive accuracy, overcoming overfitting concerns, and accommodating diverse machine learning scenarios.

3- Evaluation

Evaluation metrics for regression include R-squared, MSE, MAE, and RMSE. Those metrics can be calculated as follows:

R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables. RSS is the sum of squared residuals (model prediction errors), and TSS is the total sum of squares.

$$R^2 = 1 - \frac{RSS}{TSS}$$

MSE calculates the average squared difference between actual ($Y_i$) and predicted ($\hat{Y}_i$) values. It penalizes larger errors more heavily due to the squaring operation.

$$MSE = \frac{1}{n-(k+1)} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

RMSE is the square root of MSE, providing a measure in the same unit as the target variable. It offers an interpretable scale for the average prediction error.

$$RMSE = \sqrt{\frac{1}{n-(k+1)} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}$$

MAE calculates the average absolute difference between actual ($Y_i$) and predicted ($\hat{Y}_i$) values. It provides a more interpretable metric that is less sensitive to outliers compared to MSE.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} \left| Y_i - \hat{Y}_i \right|$$

In these formulas, n represents the number of observations in the dataset, $Y_i$ is the actual value, and $\hat{Y}_i$ is the predicted value.

Evaluation metrics for classification include Accuracy, Precision, Recall, and F1. Those metrics can be calculated as follows:

Accuracy measures the proportion of correctly classified instances out of the total instances. It provides an overall assessment of the model's correctness.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Precision quantifies the accuracy of positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives.

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

Recall assesses the model's ability to capture all relevant instances. It is the ratio of correctly predicted positive observations to the total actual positives.

$$True\ positive\ rate\ (Recall) = \frac{True\ positives}{True\ positives + False\ negatives}$$

The F1 Score is the harmonic means of precision and recall. It provides a balanced measure that considers both false positives and false negatives.

$$F1 - measure = 2 \times \frac{Precision\ \times\ Recall}{Precision +\ Recall}$$

In these formulas, "True Positives" are instances correctly identified as positive, "False Positives" are instances incorrectly identified as positive, and "False Negatives" are instances incorrectly identified as negative.

4- Normalization

Normalization is a process used in machine learning to scale numerical features to a standard range, typically between 0 and 1. One common normalization formula is:

$$x_{normalized} = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Here:

$x_{normalized}$ is the normalized value of the feature.

x is the original value of the feature.

min(X) is the minimum value of the feature in the dataset.

max(X) is the maximum value of the feature in the dataset.

This formula ensures that the feature values are linearly transformed to a range between 0 and 1, with 0 representing the minimum value and 1 representing the maximum value. Normalization helps prevent features with larger scales from dominating the learning process, especially in algorithms sensitive to the scale of input features, such as gradient-based optimization algorithms.

5- Regularization

Regularization is a technique used in machine learning to prevent overfitting by adding a penalty term to the model's cost function. For linear regression, one common form of regularization is L2 regularization (also known as Ridge regularization). The formula for the cost function with L2 regularization is:

$$J(\theta) = MSE + \lambda \sum_{i=1}^{n} \theta_i^2$$

Here:

$J(\theta)$ is the regularized cost function.

MSE is the Mean Squared Error (without regularization).

$\lambda$ is the regularization parameter, controlling the strength of the regularization.

$\theta_i$ are the model parameters.

The regularization term $\lambda \sum_{i=1}^{n} \theta_i^2$ penalizes large values of the parameters $\theta_i$. The parameter $\lambda$ determines the trade-off between fitting the data well and keeping the model parameters small. Higher values of $\lambda$ result in stronger regularization.

In the context of regularization, L1 regularization (Lasso regularization) is another common approach, and it adds the absolute values of the parameters to the cost function. The general form of the cost function with L1 regularization is:

$$J(\theta) = MSE + \lambda \sum_{i=1}^{n} |\theta_i|$$

Regularization helps to prevent the model from becoming too complex and overfitting the training data, improving its ability to generalize to new, unseen data.

6- K-Cross Validation:

The k-fold cross-validation process involves dividing the dataset into k subsets or folds. The general steps for k-fold cross-validation are as follows:

1. Dividing the Dataset: Splitting the dataset into k approximately equal-sized folds.
2. Iteration (k Times):
   a) Train-Test Split: In each iteration, one of the k folds is used as the test set, and the remaining k-1 folds are used as the training set.
   b) Model Training: Train the model on the training set.
3. Performance Metric Calculation: Evaluate the model's performance on the test set for each iteration.
4. Average Performance: Calculate the average performance metric over all k iterations.

Now, the mathematical formulas for k-fold cross-validation:

1. Test Set Index in each Iteration:
   - Test set index in iteration $i = i \; mod \; k$
   - Where $i$ is the iteration index (0 to k-1)
2. Performance Metric Calculation in each Iteration:
   - Let $Metric_i$ be the performance metric in iteration i.
   - The average performance metric (Avg_Metric) is calculated as:

$$Avg_{Metric} = \frac{1}{k} \sum_{i=0}^{k-1} Metric_i$$

This generalizes the k-fold cross-validation process mathematically. It's important to note that various performance metrics (such as accuracy, precision, recall, etc.) can be used in place of "Metric" depending on the specific evaluation criteria for the machine learning task.

# Appendix C (Default Parameters and Hyperparameters Used for ML Models)

This appendix specifies the parameters and hyperparameters of different ML models used in this research for regression and classification tasks. As mentioned in chapter 9, Scikit-Learn Python Module is used for regression and classification. This module comes with different models which are easily called using simple lines of code. The models come with default parameters and hyperparameters which were not modified.[2]

***Regression Models:***

*Random Forest Regressor:*

| Parameter | Default Value |
|---|---|
| **n_estimators** | 100 |
| **criterion {“squared_error”, “absolute_error”, “friedman_mse”, “poisson”}** | squared_error |
| **max_depth** | None |
| **min_samples_split** | 2 |
| **min_samples_leaf** | 1 |
| **min_weight_fraction_leaf** | 1.0 |
| **max_leaf_nodes** | None |
| **min_impurity_decrease** | 0.0 |
| **bootstrap** | True |
| **oob_score** | False |
| **n_jobs** | None |
| **random_state** | None |
| **verbose** | 0 |
| **warm_start** | True |
| **ccp_alpha** | 0.0 |
| **max_samples** | None |

*Linear Regression*

[2] https://scikit-learn.org/stable/modules/classes.html

| Parameter | Default Value |
|---|---|
| **fit_intercept** | True |
| **Copy_X** | True |
| **positive** | False |

*Ridge*

| Parameter | Default Value |
|---|---|
| **alpha** | 1.0 |
| **fit_intercept** | True |
| **copy_X** | True |
| **max_iter** | None |
| **tol** | 1e-4 |
| **solver {‘auto’, ‘svd’, ‘cholesky’, ‘lsqr’, ‘sparse_cg’, ‘sag’, ‘saga’, ‘lbfgs’}, default=’auto’** | auto |
| **positive** | False |
| **random_state** | None |

*k-NN*

| Parameter | Default Value |
|---|---|
| **n_neighbors** | 5 |
| **weights{‘uniform’, ‘distance’}** | unifor |
| **algorithm{‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’}** | auto |
| **leaf_size** | 30 |
| **p** | 2 |
| **metric** | ’minkowski’ |
| **metric_params** | None |
| **n_jobs** | None |

*Decision Tree*

| Parameter | Default Value |
|---|---|
| **criterion{“squared_error”, “friedman_mse”, “absolute_error”, “poisson”}** | squared_error |
| **splitter{“best”, “random”}** | best |

| | |
|---|---|
| **max_depth** | None |
| **min_samples_split** | 2 |
| **min_samples_leaf** | 1 |
| **min_weight_fraction_leaf** | 0.0 |
| **max_features {“auto”, “sqrt”, “log2”}** | None |
| **random_state** | None |
| **max_leaf_nodes** | None |
| **min_impurity_decrease** | 0.0 |
| **ccp_alpha** | 0.0 |

*XGBRegressor*

| Parameter | Default Value |
|---|---|
| **Booster Type** | gbtree |
| **Learning Task** | reg:squarederror |
| **n_estimators** | 100 |
| **learning_rate** | 0.3 |
| **max_depth** | 6 |
| **min_child_weight** | 1 |
| **Gamma** | 0 |
| **Subsample** | 1.0 |
| **colsample_bytree** | 1.0 |
| **lambda** | 1 |
| **alpha** | 0 |

***Classification Models:***

*Random Forest*

| Parameter | Default Value |
|---|---|
| **n_estimators** | 100 |
| **criterion{“gini”, “entropy”, “log_loss”}** | gini |
| **max_depth** | None |
| **min_samples_split** | 2 |
| **min_samples_leaf** | 1 |
| **min_weight_fraction_leaf** | 0.0 |
| **max_features** | sqrt |
| **max_leaf_nodes** | None |
| **min_impurity_decrease** | 0.0 |
| **bootstrap** | True |

| | |
|---|---|
| **oob_score** | False |
| **n_jobs** | None |
| **random_state** | None |
| **verbose** | 0 |
| **warm_start** | False |
| **class_weight** | None |
| **ccp_alpha** | 0.0 |
| **max_samples** | None |

*k-NN*

| Parameter | Default Value |
|---|---|
| **n_neighbors** | 5 |
| **weights :{'uniform', 'distance'}** | 'uniform' |
| **algorithm{'auto', 'ball_tree', 'kd_tree', 'brute'}** | auto |
| **leaf_size** | 30 |
| **p** | 2 |
| **metric** | Minkowski |
| **metric_params** | None |
| **n_jobs** | None |

*Decision Tree*

| Parameter | Default Value |
|---|---|
| **criterion{"gini", "entropy", "log_loss"}** | gini |
| **splitter** | best |
| **max_depth** | None |
| **min_samples_split** | 2 |
| **min_samples_leaf** | 1 |
| **min_weight_fraction_leaf** | 0.0 |
| **max_features** | None |
| **max_leaf_nodes** | None |
| **min_impurity_decrease** | 0.0 |
| **random_state** | None |
| **class_weight** | None |
| **ccp_alpha** | 0.0 |

*AdaBoost*

| Parameter | Default Value |
|---|---|

| | |
|---|---|
| **estimator** | None |
| **n_estimators** | 50 |
| **learning_rate** | 1.0 |
| **algorithm : {‘SAMME’, ‘SAMME.R’}** | SAMME.R |
| **random_state** | None |
| **base_estimator** | None |

*SVC*

| Parameter | Default Value |
|---|---|
| **penalty{‘l1’, ‘l2’}** | l2 |
| **loss{‘hinge’, ‘squared_hinge’}** | squared_hinge |
| **dual** | True |
| **tol** | 1e-4 |
| **C** | 1.0 |
| **multi_class{‘ovr’, ‘crammer_singer’}** | ovr |
| **fit_intercept** | True |
| **intercept_scaling** | 1.0 |
| **class_weight** | None |
| **verbose** | 0 |
| **random_state** | None |
| **max_iter** | 1000 |

*Conversation with ChatGPT v3.5 to brainstorm the form making of an exhibition for Mercedes-Benz.*

***Q1: Architect***: Imagine that you are my architectural design partner to discuss and form a concept of a new design project.

***ChatGPT***: Certainly! I'm excited to collaborate with you on your architectural design project. To get started, could you provide some initial details about the project? What type of building or space are we designing, and are there any specific requirements or preferences you have in mind? Additionally, do you have any particular themes, styles, or inspirations you'd like to explore in the design?

***Q2: Architect***: We are designing a car exhibition for Mercedes-Benz. The project will have a wide entrance, exhibition spaces with many forms of show like physical, VR, holograms, video shows, etc., as well as some functional spaces like restrooms, cafeteria, and administration spaces.

I am thinking of a contemporary design style that should symbolize and express the core concepts of the firm (Mercedes-Benz). And right now, we need to brainstorm the form-making process.

***ChatGPT:*** Great! Designing a car exhibition for Mercedes-Benz sounds like a fantastic project. Let's dive into the form-making process and explore some concepts for the overall design. Since we want to capture the essence of Mercedes-Benz and convey their brand identity, here are a few initial ideas to consider:

- Fluid Lines and Elegance: Incorporate fluid lines and elegant curves in the architecture to reflect the sleek and dynamic design of Mercedes-Benz cars. Consider using materials with reflective surfaces to evoke the polished finish often found in luxury cars.
- Showcase of Innovation: Integrate cutting-edge technology into the design to reflect the innovation associated with Mercedes-Benz. Use LED displays, interactive touchscreens, and advanced lighting systems to create an immersive and futuristic experience.

- Spatial Hierarchy: Design the exhibition spaces with a clear hierarchy, emphasizing key car models and innovations. Create focal points within the exhibition area to draw attention to flagship vehicles or groundbreaking technologies.
- Branding Elements: Integrate Mercedes-Benz branding elements into the architecture, such as the iconic logo, in a subtle and sophisticated manner. Explore the use of branded colors and textures that align with the company's visual identity.
- Multi-sensory Experience: Incorporate a variety of sensory experiences, including virtual reality (VR), holograms, and dynamic video displays to engage visitors on multiple levels.
- Consider the use of ambient sounds that evoke the sensation of being on the road or in a high-tech automotive environment.
- Green Design: Infuse sustainable design principles into the project, aligning with the growing emphasis on eco-friendly practices. Consider incorporating greenery and natural elements to create a harmonious and inviting atmosphere.
- Spatial Flexibility: Design flexible exhibition spaces that can adapt to different types of showcases, ensuring versatility for various events and car presentations.

Let's discuss these concepts further and refine them based on your preferences. Are there any specific aspects you'd like to emphasize or any particular features you envision for the exhibition space?

***Q3: Architect***: Yes, we need to dive deeper in the form-making aspects. How should the building look like, and I am thinking to have an analogy from nature that expresses Mercedes-Benz codes and concepts

***ChatGPT:*** Certainly, delving deeper into the form-making aspects and drawing inspiration from nature can result in a unique and meaningful design. Let's explore the analogy from nature to express Mercedes-Benz codes and concepts. Here are a few nature-inspired ideas:

- Flowing Lines and Organic Forms: Take inspiration from the streamlined shapes found in nature, such as the curves of a river or the organic forms of plant leaves. Consider incorporating these flowing lines into the architecture, creating a sense of movement and elegance that mirrors the design philosophy of Mercedes-Benz.

- Biomorphic Architecture: Explore biomorphic architecture, which mimics the forms and structures found in living organisms. Use organic shapes and patterns reminiscent of natural elements, providing a visual connection to the brand's emphasis on innovation and evolution.

- Tree-Inspired Canopy: Design a canopy structure inspired by the branching and interwoven patterns of trees. This canopy can serve as a unifying element over the entrance and exhibition spaces, providing shade and a sense of cohesion while symbolizing growth and strength.

- Reflective Surfaces Inspired by Water: Draw inspiration from the reflective qualities of water surfaces, like a calm lake or a flowing stream. Use reflective materials on certain surfaces to create a dynamic play of light, enhancing the sense of luxury and sophistication.

- Geometric Patterns Inspired by Crystals: Consider incorporating geometric patterns inspired by the precision and clarity of crystals. These patterns can be integrated into the design of structural elements or even as artistic installations, emphasizing the precision and craftsmanship associated with Mercedes-Benz.

- Landscaped Courtyards: Create landscaped courtyards with carefully selected vegetation, emphasizing a connection to nature. These green spaces can serve as areas for relaxation and reflection, providing a contrast to the high-tech exhibition spaces.

- Dynamic Facade Inspired by Wind Patterns: Look to wind patterns as inspiration for a dynamic and responsive facade. Design elements that move or adjust based on environmental conditions, symbolizing adaptability, and responsiveness—qualities associated with Mercedes-Benz.

As we explore these nature-inspired concepts, please feel free to provide feedback or let me know if there's a particular direction you find intriguing. We can further refine the ideas to ensure they align seamlessly with the vision for the Mercedes-Benz exhibition space.

***Q4: Architect***: instead of expressing the precision and craftsmanship using Crystal patterns, let's suggest another pattern that expresses Mercedes-Benz sustainability.

Absolutely, let's explore a pattern inspired by sustainability for expressing precision and craftsmanship. One compelling option could be the use of "Geometric Patterns Inspired by Sustainable Materials."

Consider incorporating patterns reminiscent of sustainable materials like bamboo, recycled wood, or even recycled metals. These materials can be translated into geometric patterns that symbolize environmental responsibility and the commitment to sustainability. The repetition of these patterns can create a visually striking and cohesive design element throughout the exhibition space.

This not only aligns with the sustainability aspect but also emphasizes precision in the use of materials and craftsmanship in creating a harmonious design. Additionally, using sustainable materials in the construction of the exhibition space reinforces Mercedes-Benz's commitment to environmental consciousness.

How does this idea resonate with you, and is there a specific sustainable material or concept you'd like to emphasize within the geometric patterns?

***Q5: Architect***: How could the geometry inspired by bamboo look like?

***ChatGPT:*** Designing geometry inspired by bamboo can add a natural and sustainable aesthetic to the exhibition space. Here are some design elements and principles that could be incorporated:

Vertical Rhythm: Mimic the vertical rhythm found in bamboo stalks. Use vertical lines or columns to create a sense of height and rhythm throughout the space.

These vertical elements can be part of the structural design, interior partitions, or even incorporated into artistic installations.

Segmentation and Joints: Emulate the segmented structure of bamboo stalks with distinct joints. These joints can be represented through the use of varying materials, textures, or color changes.

Consider using contrasting materials to highlight these joints, showcasing the precision and craftsmanship in the design.

Natural Curves and Tapers: Integrate gentle curves and tapers reminiscent of bamboo stalks. This can be applied to the layout of pathways, the curvature of walls, or even in the design of furniture and fixtures. These natural curves add a sense of fluidity and organic beauty to the space.

Latticework Patterns: Incorporate latticework patterns inspired by bamboo weaving techniques. These patterns can be applied to screens, partitions, or as decorative elements within the exhibition spaces. The latticework not only adds a visual appeal but also allows for the play of light and shadow, creating a dynamic and engaging atmosphere.

Natural Color Palette: Draw inspiration from the natural colors of bamboo, such as earthy greens and browns. Use a palette that reflects the warmth and sustainability associated with natural materials. Incorporate these colors into furnishings, finishes, and decorative elements throughout the space.

Modularity and Flexibility: Take inspiration from bamboo's modularity and flexibility in design. Create modular elements that can be rearranged or repurposed, promoting adaptability and sustainability in the use of space.

By integrating these elements, the exhibition space can evoke the essence of bamboo while emphasizing precision and sustainability. This design approach not only aligns with Mercedes-Benz's commitment to environmental responsibility but also creates a visually captivating and cohesive experience for visitors. How does this concept resonate with your vision for the project?

***Q6: Architect***: how can we adopt such concept to le Corbusier’s regulating lines?

***ChatGPT***: Combining the concept of bamboo-inspired geometry with Le Corbusier's regulating lines involves incorporating the principles of both into the design. Le Corbusier's regulating lines, based on his Modulor system, emphasize human proportions and geometric harmony. Here's how you can integrate these two concepts:

Vertical Modules Inspired by Bamboo: Use Le Corbusier's modular system to establish vertical regulating lines based on human proportions. Align the vertical rhythm of the regulating lines with the vertical segments of the bamboo-inspired design. This creates a sense of harmony and proportion within the space.

Human-Scale Integration: Leverage the Modulor system to determine the scale of various elements within the exhibition space, such as display areas, partitions, and seating.

Apply the bamboo-inspired geometry within these modules, ensuring that the natural elements align with the human-scale proportions defined by Le Corbusier.

Segmented Joints within Regulating Lines: Integrate the segmented joints inspired by bamboo within the regulating lines. This creates a visually cohesive and harmonious relationship between the precision of Le Corbusier's geometric principles and the organic, sustainable elements inspired by bamboo.

Curvilinear Forms Aligned with the Modulor: Apply Le Corbusier's principles of modularity to introduce curvilinear forms within the exhibition space. Incorporate the natural curves and tapers inspired by bamboo, ensuring that these organic elements align with the modular scale defined by Le Corbusier.

Latticework Patterns within Proportional Modules: Utilize Le Corbusier's proportional modules to guide the placement and design of latticework patterns inspired by bamboo. Ensure that the latticework patterns align with the regulating lines and modular grid, reinforcing a sense of order and proportionality.

Natural Color Palette and Modulor Harmony:

Integrate the natural color palette inspired by bamboo within the Modulor system. Harmonize the earthy greens and browns with the proportional modules, creating a balanced and visually appealing color scheme.

By integrating the bamboo-inspired geometry within the framework of Le Corbusier's regulating lines, you create a design that marries organic, sustainable elements with precise, human-centered proportions. This approach not only aligns with the principles of both concepts but also results in a sophisticated and harmonious exhibition space for Mercedes-Benz. How does this synthesis of ideas resonate with your vision for the project?

جامعة عين شمس
كلية الهندسة
قسم الهندسة المعمارية

# ترسيم عملية الفكرة المعمارية من خلال توظيف الذكاء الاصطناعي في الوصول إلى التشكيل

رسالة مقدمة من
**عبد الرحمن أيمن فهمي سالم**
ماجستير العلوم في العمارة 2019 – جامعة عين شمس
بكالوريوس الهندسة المعمارية 2015 – جامعة عين شمس

تحت إشراف:

**أ.د. ياسر محمد منصور**
أستاذ نظريات العمارة
جامعة عين شمس

**أ.د. حازم طلعت الدالي**
أستاذ نظريات العمارة
جامعة عين شمس

2024

# لجنة الحكم والمناقشة

**الاسم:** عبد الرحمن أيمن فهمي سالم

**عنوان الرسالة:** ترسيم عملية الفكرة المعمارية من خلال توظيف الذكاء الاصطناعي في الوصول إلى التشكيل

**الدرجة:** دكتوراه الفلسفة في العمارة

**تاريخ مناقشة البحث:**

| عضو لجنة الحكم والمناقشة | التوقيع |
| --- | --- |
| **أ.د. سمير صادق حسني**<br>أستاذ نظريات العمارة بكلية الهندسة جامعة .عين شمس | |
| **أ.د. ياسر محمد منصور**<br>أستاذ نظريات العمارة بكلية الهندسة جامعة عين شمس (مشرفاً) | |
| **أ.د. .شريف مراد عبد القادر**<br>أستاذ العمارة بكلية الهندسة بالجامعة الأميريكية | |
| **أ.د. حازم طلعت الدالي**<br>أستاذ نظريات العمارة بكلية الهندسة جامعة عين شمس (مشرفاً) | |

---

**الدراسات العليا:**

أجيزت الرسالة بتاريخ: / /

ختم الإجازة　　　　موافقة مجلس الكلية:　　　　موافقة مجلس الجامعة:

# ملخص البحث

لقد أدت التطورات الســـريعة في التعلم الآلي إلى العديد من التطبيقات العملية في مختلف المجالات. وقد بدأ المهندسـون المعماريون والباحثون أيضًـا في اسـتكشـاف إمكانات تعلم الآلة لتعزيز عملهم. ومع ذلك، غالبًا ما تفشل التطبيقات الحالية في توفير نماذج معمارية دقيقة وسهلة الاســتخدام ضــمن برامج التصــميم والرســم المتعارف عليها والتي يســتخدمها المهندســون المعماريون. ولمواجهة هذا التحدي، نقدم في هذا البحث نهجًا جديدًا يعزز هندســــة الترميز في عملية خوارزمية لترجمة المعلومات المعمارية إلى أنواع بيانات يمكن فهمها بواسطة الآلة، مثل الأرقام الكسرية والأرقام الصحيحة والنصوص. يبدأ المسار المقترح من تصميم فيلا بارامترية باســتخدام برنامج Rhinoceros3d وباســتخدام لغة البرمجة C#، وذلك لإنشــاء مجموعة بيانات كبيرة عن طريق تغيير المعاملات (المتغيرات)، ثم تدريب نماذج تعلم الآلة باســــتخدام مجموعة البيانات. يشــمل النموذج البارامترى الذي تم إنشــاؤه مجموعة واســعة من المعاملات المترابطة، بما في ذلك أبعاد الحوائط، وارتفاع الأرضــــيات، وأبعاد التجاويف، وخصــــائص النوافذ، ومســـاحة البناء، وأبعاد الأرض. يتم تنفيذ النموذج بأكمله باســـتخدام لغة البرمجة RhinoCommon API وC#. يسهل النموذج البارامترى الناتج التخزين التلقائي للبيانات في ملف CSV منســـق لاســـتخدامه مباشـــرة في تعلم الآلة. وتم في نطاق البحث اختبار خوارزميات تعلم الآلة المختلفة على أربع مجموعات من البيانات والتي تم إنشاؤها من النموذج وهي مجموعة بيانات للتنبؤ بالمعاملات المتعلقة بالمناطق، وواحدة للتنبؤ بالمعاملات المتعلقة بمعلمات النموذج الأخرى، وواحدة للتنبؤ بوجود النوافذ في كل جدار، وأخرى للتنبؤ بعرض النوافذ. تتطلب مجموعات البيانات خوارزميات الانحدار والتصـــنيف للتنبؤ بجميع المعاملات. وقد نتج عن التجربة تحقيق نتائج جيدة باســـتخدام طرق التعلم المجمعة مع جميع مجموعات البيانات. وقد وصـــلت نتائج مهام الانحدار إلى درجة R2 تصـــل إلى 0.97 و0.79 و0.99 للمناطق والمعاملات الأخرى ومجموعات بيانات عرض النوافذ ودقة تصل إلى 98% في مهمة تصـنيف وجود النوافذ. وقد تم حسـاب جميع النتائج على مجموعة بيانات الاختبار. تسـلط هذه النتائج الضـــوء على فعالية نهجنا في توليد تنبؤات معمارية دقيقة من خلال تقنيات التعلم الآلي والتي تعد ذات أهمية بالغة في عملية أتمتة النمذجة المعمارية.

**<u>الكلمات المفتاحية:</u>** الصــندوق الأســود، التفكير التصــميمي المعماري، الذكاء الاصــطناعي، التصميم التوليدي، الوصول إلى التشكيل، التعلم الآلي، الانحدار، التصنيف، الشبكات العصبية، الترميز، البرمجة.

## 1.تمهيد

تبين أن أساليب التصميم المعمارية التي يستخدمها المهندسون المعماريون عادة هي الأكثر غموضاً. وتشمل هذه الأساليب طريقة تصميم الصندوق الاسود حيث يولد مفهوم التصميم في عقل المهندس المعماري وحده. وبالطبع، فإن الكثير من المهندســـين المعماريين يريدون أن

يكونوا فريدين من نوعهم، وكل المهندسين المعماريين يحصلون على لغتهم ومفرداتهم الخاصة في الهندسة المعمارية التي ينظر إليها بطريقة ما على أنها متكررة وتعبر عن المهندس المعماري نفسه. ومع ذلك، فإن عملية التصميم لا تكون أبداً خطية. وهي عملية دورية تنشأ فيها الفكرة المعمارية للدخول في حلقة من التطورات والتحريرات من خلال تحليل المهندس المعماري في كل مرحلة وبعد كل خطوة. وهذه العملية الدورية تجعل من الصعب شرح عملية التصميم.

ومن ناحية أخرى، أثر ظهور الحقبة الرقمية على العديد من مجالات الفن والعلوم على حد سواء، بما في ذلك الهندسة المعمارية. في البداية، تم التفكير في التصميم على الأوراق عن طريق الرسومات. وفي الوقت الحاضر، تستخدم مئات البرامج الحاسوبية كأدوات لمساعدة المهندس المعماري على ترجمة أفكاره الخاصة التي من المؤكد أنها أصبحت أكثر تعقيداً مع المواد الجديدة وتقنيات التصنيع التي سمحت بسهولة بناء المباني ذات الشكل الحر. لم تتوقف التكنولوجيا فقط في مساعدة المهندسين المعماريين للتعبير عن أنفسهم، ولكن حدثت قفزة في المجال المعماري عندما تم إدخال التصميم التوليدي. والتصميم التوليدي يمكن وصفه بأنه عملية توليد الآلة عددا معينا من النواتج (التصميمات) استنادا إلى البيانات التي تُغذى بها. وقد شهد مجال التصميم المعماري تغيراً في دور المعماري بعد هذه القفزة حيث تحول من تصميم بديلين أو ثلاثة لمبنى معين يستند إلى قيود ومتغيرات مختلفة ومعروفة إلى تزويد الآلة بتلك القيود والمتغيرات ومجموعة من الخوارزميات لتوليد مئات وربما الآلاف من البدائل بل وتصنيفها لمساعدة المهندس المعماري على اختيار أفضل البدائل من وجهة نظره وذلك عن طريق تحقيق حل المشاكل والقيود.

وقد شهد هذا التغيير في الدور الكثير من النقد من المصممين الذين يعتقدون أن تصميم بديلين أو ثلاثة بدائل مدروسة جيداً يمكن أن تكون أكثر كفاءة ومهنية من الآلاف من التصميمات غير المدروسة. ليس فقط التأثير على دور المهندس المعماري الذي تعرض للانتقاد، ولكن أيضاً الوقت الذي تستغرقه مرحلة صنع القرار لاختيار أفضل مجموعة من البدائل من الآلاف منها.

الآن، ماذا لو تم تغذية الآلة بالمتغيرات، القيود، والعلاقات المناسبة - والتي تعبر أيضاً عن طريقة المعماري في التفكير - من خلال خوارزميات التي يمكن أن تجعل الآلة تولد أشكال مقبولة ومدروسة جيداً بدلاً من توليد الآلاف؟ وعلاوة على ذلك، ماذا لو طُبقت تكنولوجيات الذكاء الاصطناعي والتعلم الآلي بحيث يمكن للآلة أن تتعلم من المعماري استناداً إلى خياراته وقراراته ويمكنها أن تكون قادرة على التفكير مثل الشخص الذي يزودها بالبيانات للحصول على نتائج محددة؟

في الوقت الحاضر، حقق الذكاء الاصطناعي نجاحاً في العديد من الميادين مع العديد من الأدوات المتاحة لعامة الجمهور لاستخدامها. كما أن المعرفة التي تستند إليها أنظمة الذكاء الاصطناعي هي أكثر انتشاراً ويمكن وصول الجميع إليها واستخدامها بشكل مباشر في مختلف التطبيقات. في الهندسة المعمارية، أجريت العديد من التجارب مع تطبيقات AI وتطبيقات

MLلسنوات عديدة، الآن. وأفضى ظهور هذا البرنامج إلى إتاحة فرص جديدة للمهندسين المعماريين الذين بدأوا باستخدام نماذج توليد الصور للمساعدة في مرحلة وضع الفكرة المعمارية، فضلاً عن المشاريع التصورية، ووضع الخطط، وما إلى ذلك. كما انتشرت اليوم تطبيقات إنتاج الفيديو والنماذج. ومع ذلك، تستخدم نماذج الذكاء الاصطناعي التوليدية نماذج مثل point clouds، أو voxels، أو نماذج NERF لتوليد نماذج ثلاثية الأبعاد. وهذه النماذج لا تولد نماذج معمارية قابلة للتطوير. وعلاوة على ذلك، أُجري المزيد من التجارب على التطبيقات غير المولِّدة للابتكار في مجالات التخطيط، والتنبؤ بالمواد، والتصنيف، وغيره.

يهدف هذا البحث للإجابة عن الأسئلة السابقة من خلال دراسة استخدام AI وML في التصميم المعماري في الوصول إلى التشكيل. وهذا يتطلب دراسة متعمقة للأسلوب المعماري المستخدم من أجل ترجمة معاملاته ومحركاته إلى متغيرات وقيود مفهومة يتعين تغذية الآلة بها لتوليد تصاميم تعبر عن المهندس المعماري والمنتج النهائي المرغوب فيه. وبالإضافة إلى ذلك، يهدف البحث إلى اختبار استخدام الترميز في هذا التطبيق نظراً لأن الترميز يُنظر إليه على أنه يضيف الكثير إلى هذا الإطار في التصميم المعماري والذي يستخدم تطبيقات مختلفة من تعلم الآلة في عملية النمذجة. كما أن البحث يختبر العديد من نماذج ML للحصول على معلومات عن النماذج التي تناسب هذه التطبيقات بشكل جيد.

## 2.المشكلة البحثية

إن التصميم التوليدي -مهما بدى مميزاً ومؤثراً في عملية التصميم المعماري- إلا أنه يصاحبه بعض المشاكل وخصوصاً في مراحل اتخاذ القرار. في هذه العملية يتم توليد آلاف الحلول والبدائل التصميمية والتي يمكن الاختيار من بينها للتطوير. ومع هذا، هناك العديد من البدائل لا يمكن الاعتماد عليها بناءً على عوامل عديدة مؤثرة مثل قوانين البناء، الشكل المعماري، بيئة الأفراد، العلاقات بين الفراغات، او حتى تفضيلات المعماري الجمالية والفنية والتي تخص تكوين الكتلة المعمارية.

بهذا الصدد، نرى أن كثير من المعماريين يستهلكون الكثير من الوقت لتنظيم وترتيب أفضل البدائل المولدة ليتم دراستها وتطويرها. بالإضافة لذلك، فإنه كلما كثرت المعاملات التي يتم تحديدها لطرق التصميم التوليدي، كلما زادت الأفكار الناتجة والتي تؤدي لاستهلاك وقت أكبر في اتخاذ القرارات.

كذلك، تحتاج الآلة للكثير من المعلومات لتستطيع بنفسها تحليل وفرز النتائج الأفضل والتي تحقق ما يتطلبه المشروع بالإضافة إلى ما يعبر عن شخصية المعماري والمتمثلة في صندوق الأفكار الأسود حتى تستطيع تحسين النتائج ويتمكن المعماري من الاعتماد عليها في اتخاذ القرارات لتقليل عدد النتائج المولدة والتي تتميز بقيمة تصميمية أفضل بدلاً من عدد أكبر من البدائل.

إن الوقت المستغرق في تصميم المبنى يأخذ الكثير من ساعات العمل. حتى تصميم المزيد من النماذج الأولية للمبنى سوف يستغرق الكثير من الوقت على الرغم من أن الأسلوب قد يكون قد تم تحديده بالفعل .

وللاستفادة من AI في ادخار الوقت، توجد بعض النماذج التوليدية التي تولد صوراً (التصميمات؟) من المخططات والمناظير ويستخدمها المعماريون على نطاق واسع في الوقت الحاضر. ومع ذلك، تأتي مثل هذه التطبيقات مع العديد من القضـــايا المتعلقة بصـــحة، وإبداع، وحتى وظيفة النتيجة التي نعتقد أنه لا ينبغي حتى اعتبارها منتجا معماريا.

وأخيراً، وبغرض أتمتة بعض المهام في مرحلة التصميم، تم تجريب بعض تطبيقات التصنيف والتراجع للتنبؤ أو تصنيف المواد، والبارامترات المعمارية، وما إلى ذلك. ومع ذلك، لا يمكننا أن نتتبع أي تجارب لإنشاء نماذج معمارية من خلال نظام تعلم الآلة يمكن استخدامها مباشرة في مراحل أخرى مثل التصميم التخطيطي وتطوير التصميم.

وعلى الرغم من أن مجال الذكاء الاصـــطناعي وتعلم الآلة يعتبران اليوم متقدمين جداً، فإن هناك العديد من المفاهيم في هذا الميدان بحاجة إلى إعادة النظر فيها من جانب المهندسـين المعماريين لتسـخير قوة الآلة في تشغيل المهام التي تستغرق وقتاً طويلاً.

## 3.فرضية البحث

إن طريقة تفكير المعماري تُعتبر عملية يمكن تتبعها في مرحلة التفكير التصـميمي. ويمكن ترجمة هذه العملية، مقرونة بمفردات الأسلوب المعماري، إلى معاملات تتجمع لتؤدي إلى منتج يشـبه لما يحدث في أذهان المهندس المعماري في هذه العملية. وإذا ما درسـت هذه المعاملات دراسة جيدة وأدخلت بشكل جيد إلى الآلة، يمكن للآلة أن توفر بدائل توليدية مناسبة ودقيقة يمكن أن تقلل من الوقت المسـتهلك في عملية صـنع القرار. وبالإضـافة إلى ذلك، يمكن للآلة أن تتعلم من خيارات المهندس المعماري ومن ثم تعطي بدائل أكثر دقة وأكثر صـلة بالموضـوع إما في المرحلة التالية أو حتى في المشـروع التالي. ويمكن لآلية التعلم هذه أن تؤدي إلى عهد جديد من المفاهيم المعمارية التفاعلية البشرية - الآلية.

## 4. الأهداف الرئيسية للبحث

يهدف هذا البحث لترسيم طراز العمارة المعاصـرة بمعاملاته ومرادفاته المتعلقة بالتشكيل، واستخدام التصميم التوليدي لتوليد تصميمات مختلفة مع توظيف الذكاء الاصطناعي وتعلم الآلة لتتعلم من القرارات التي يتخذها المصـمم مما يقلل الاحتمالات التصـميمية الناتجة بالتعلم والذي تأخذ الآلة فيه شـخصـية المعماري مما يولد أفكاراً تصـميمية مشـابهة لما قد ينتج منه باسـتخدام الطريقة التقليدية في التصميم.

## 5.الأهداف الفرعية للبحث

يمكن تحقيق الأهداف الرئيسية من هذا البحث من خلال الأهداف الفرعية التالية:

- تعريف عملية التصميم المعماري التقليدية وأساليبها المختلفة.

- تعريف بعض الاســتعارات المســتخدمة في التفكير المعماري، بما في ذلك الصــندوق الاسود، الصندوق الابيض، والصندوق الرمادي.
- تحليل الأسلوب المعماري المعاصر.
- ترسيم مفردات وعناصر أسلوب العمارة المعاصر.
- ترجمة مفردات الأسلوب المعماري إلى معاملات يمكن استخدامها في أساليب التصميم التوليدي الحديثة.
- تطبيق بعض العلاقات بين البارامترات من أجل تشــكيل نماذج توليدية عن طريق الترميز.
- تحليل تطبيقات الذكاء الاصــطناعي وتعلم الآلة لفهم قدراتهما وتقرير كيفية الاســتفادة منهما.
- وضع إطار لصنع القرار في النمذجة المعمارية باستخدام تعلم الآلة.
- اســتخدام تقنيات تعلم الآلة في وضــع تصــميمات ذات صــلة بطابع/أسـلوب/خيارات/أفضـليات المهندس المعماري/المشـروع والخيارات والأفضـليات السابقة التي اتخذها المهندس المعماري في هذا الصدد.

## 6. نطاق البحث

هذا البحث موجه نحو استقصاء نماذج تعلم الآلة تحديداً من مجموعة نماذج الذكاء الاصطناعي. ويمكن لنماذج تعلم الآلة أن تتنبأ أو تصــنف أو تجمع البيانات اســتناداً إلى مجموعات البيانات التي تتعلم منها. لذلك فهي لا تنتج بيانات جديدة ولكن تستطيع التعميم على البيانات غير المرئية بناء على مفاهيم رياضية من الجبر الخطي، الأســاليب الرقمية. ومع ذلك، يجري التحقيق في طائفة أوســع من نماذج الذكاء الاصطناعي لفهم وتحليل الاختلافات بين المفاهيم والتطبيقات على حد سواء.

وبالإضافة إلى ذلك، بما أن نتيجة هذا النوع من التطبيقات يمكن اعتبارها " تصميماً مبتكراً "، فإن تحليل كيفية عمل نظم التصميم التوليدي ضروري للحصول على معلومات عن الكيفية التي يمكن أن تطبق بها أنظمة تعلم الآلة المختلفة.

وللعمل على تعلم الآلات، يوصــى بأن يتم بناء نمذجة التصــميم المعماري من خلال الترميز، وبالتالي، فإن البحث موجه أيضــا نحو إجراء دراســة شــاملة عن كيفية تســخير القوة وحرية الترميز لتوليد نماذج ثلاثية الأبعاد. ولهذا الغرض، تم تصــميم فيلا معاصــرة وتُم نمذجتها من خلال الترميز باســتخدام لغة البــرمــجــة C# فــي تــطــبــيــق Grasshopper 3dبــبــرنــامــج rhinoceros 3d باستخدام.RhinoCommon API

والنتيجة من هذه الخطوة الأولى هي الحصول على مجموعة بيانات تحتوي على قيم معاملات لــــ 600 نموذج أولي للفيلا المعاصرة استنادا إلى بعض الحالات ذات الصلة بإجمالي مساحة البناء التي تتراوح بين 200 و1000 متر مربع، و الجار اذا ما كان شـــارع او مبنى، وأبعاد الأرض، وغير ذلك من السيناريوهات. وقد اختيرت الفيلا ليكون ذات أسلوب " معاصر " لتسهيل عملية البناء والترميز حيث إن الغرض الرئيســي هو إيجاد علاقة بين ما يقرب من مائة من معاملات الفيلا، وتدريب نظام تعلم الآلة واختبار إذا كان يمكن العثور على نمط بين المعاملات. وللسـبب نفسـه، لم يدرج موقع الفيلا (البلد) في المعادلة.

وقد اختبرت خوارزميات مختلفة من خوارزميات تعلم الآلة والشـبكات العصـبية الاصـطناعية لكل من مهام التصنيف والانحدار التي أسندت إما للتنبؤ بقيم البارامترات أو لتصنيف بعض البارامترات وتحقيق هدف البحث.

## 7. منهج البحث

يتم اتباع منهجيات متتالية في هذا البحث لتحقيق الهدف العام منه والأهداف الجزئية، وتشـــمل هذه المنهجيات التالي:

| *المنهجية* | *التطبيق* |
|---|---|
| *التحليل النقدي* | تعريف التفكير وحل المشكلة |
| | تعريف التفكير المعماري |
| | فهم التعقيدات المختلفة في التصميم المعماري |
| | تحليل عملية التصميم المعماري وتكوين الكتلة |
| | ترسيم الأنماط والأشكال المستخدمة في التصميم المعاصر |
| | تعريف الذكاء الاصطناعي وتعلم الآلة وتطبيقاتهما وأنواعهما |
| *دراسات تحليلية* | تحليل اســـتخدامات الذكاء الاصـــطناعي وتعلم الآلة في التصـــميم المعماري |
| *تجارب عملية* | تحويل العناصر المعمارية لعوامل يتم استخدامها في البرمجة. |
| | كتابة برنامج للتصميم التوليدي من خلال البرمجة. |
| | إنتاج معلومات عن معاملات التصميم يمكن استخدامها لتمرين أنظمة تعلم الآلة |
| | تحديد المشكلات ومعرفة أفضل أنظمة تعلم الآلة لحلها |

| | |
|---|---|
| | تطبيق أنظمة تعلم الآلة لتستطيع الآلة أن تتعلم من المعماري عن طريق ترسيم البيانات والتنبؤ أو التصنيف للمعاملات المتعلقة بالمبنى تحت الدراسة |
| | تطبيق الترميز لتعديل النموذج ثلاثي الأبعاد الناتج عن الهيكل المقترح ليستطيع المعماري التدخل في مرحلة التعليم ومرحلة تطوير النتيجة بطريقة سهلة. |

## 1- هيكل البحث

يرتبط هيكل البحث بمنهجه في توزيع أجزائه وأبوابه:

**الجزء الأول: عملية التصميم المعماري من الإنسان إلى الآلة**

يتناول الجزء الأول من البحث تحليلاً لمفاهيم مرتبطة بعملية التصميم المعماري والتفكير المعماري وكيف تحولت اليوم للاعتماد على الآلة بشكل كبير في اتخاذ القرارات.

*الباب الأول: التفكير التصميمي المعماري: الصندوق الأسود أم الصندوق الزجاجي*

في الباب الأول يتم تعريف التفكير ومفهوم حل المشكلة ومفهوم التفكير التصميمي المعماري. كما يتم عرض أنواع التصميم وبعض الاستعارات المتعلقة بالتصميم والتفكير مثل الصندوق الأسود والأبيض والرمادي. ويتم عرض التعقيدات الملازمة لمجال التصميم المعماري وعرض كيفية تقييم النتاج المعماري. يتعرض الباب الأول لعملية التصميم المعماري وبعض طرقها بالإضافة لما وصلت إليه العملية اليوم. ويتم شرح تأثير التكنولوجيا الحديثة في هذا المجال.

*الباب الثاني: ترسيم مكونات التشكيل في العمارة*

في هذا الباب، يتم تعريف التكوين المعماري وخصائصه وتمييز الفرق بين أنواع توليد التشكيل والتي تشكل تكوين التشكيل والوصول إلى التشكيل. وفي نهاية الباب، يتم عرض وتحليل بعض الأنماط والعناصر المتكررة في العمارة المعاصرة.

*الباب الثالث: الترميز في التصميم الحوسبي كقاعدة لاستخدام الذكاء الاصطناعي في الوصول للتشكيل*

في الباب الثالث يتم تعريف التفكير التصميمي الحسابي وكيفية نشأته. كما يتم التعريف ببعض المصطلحات التي نتجت عن اتباع هذا النهج والتفرقة بينها. ويركز الباب على توليد

الوصــول للتشــكيل. ثم يتم التعمق في التفكير في التكوينات المعمارية كمعلومات واسـتخدام الخوارزميات والمعاملات في التصميم المعماري.

كذلك يتعرض هذا الباب لمفهوم الترميز وكيفية عمل برامج النمذجة الحاســوبية. ثم يتم شرح مفهوم لغات البرمجة المرئية والتحيزات المصاحبة لعملية النمذجة المعمارية. ويطرح الباب تطبيق الترميز في نمذجة التكوينات المعمارية للاستفادة بالقوة والحرية المصاحبة لهذه العملية.

**<u>الجزء الثاني: تكامل الترميز وتعلم الآلة والذكاء الاصـــطناعي مع عملية التصـــميم المعماري (هيكل لاستخدام الذكاء الاصطناعي في إنشاء التشكيل)</u>**

يركز الجزء الثاني على عرض اســـتخدام الترميز في النمذجة ثلاثية الأبعاد في العمارة بالإضافة لشرح مفاهيم وأنواع الذكاء الاصطناعي وأنظمة تعلم الآلة المختلفة واستخداماتها الحالية والمحتملة في مجال التصـــميم المعماري وتكوين التشـــكيل المعماري. كما يطرح الجزء الثاني هيكل عملي لاستخدام تقنيات تعلم الآلة في عملية الوصـول للتشـكيل ونمذجة التكوينات المعمارية.

*<u>الباب الربع: الذكاء الاصطناعي وتعلم الآلة في مجال العمارة</u>*

في الباب الرابع، يتم تعريف الذكاء الاصطناعي وتعلم الآلة وتاريخ نشأة هذا العلم بالإضافة لأنواع كلا التخصـــصـــين. ويتطرق الباب للبيانات التي يتعامل بها تعلم الآلة وبعض الخوارزميات كتطبيق لهذا المجال. وينتهي الباب بشـــرح بعض طرق تقييم أداء أنظمة تعلم الآلة المختلفة. كما يتم في هذا الباب يتم عرض العديد من اســـتخدامات أنظمة الذكاء الاصطناعي التوليدية وغير التوليدية في العمارة والمقارنة بينها وتقييمها.

يطرح الباب الرابع أيضاً تشكيكاً في قدرات أنظمة الذكاء الاصطناعي التوليدية في عملية التصميم المعماري لمخاوف من ضمنها أصالة النتيجة. ويتناول الباب بعض الاستخدامات المقترحة لكلا النوعين من أنظمة الذكاء الاصطناعي (التوليدية وغير التوليدية).

*<u>الباب الخامس: توليد التشـــكيل المعماري: تطبيق خوارزميات تعلم الالة على بيانات البارامترات المعمارية</u>*

يتم تحديد المشـكلة ونطاق التجربة والمواد والطرق المسـتخدمة بها في هذا الباب. كما يتم شـــرح كيفية ترميز النموذج ثلاثي الأبعاد للمبنى تحت التجربة وكيفية اســـتخراج جداول بيانات ملائمة لاســـتخدامها في أنظمة تعلم الآلة المختلفة. كذلك يتم شـــرح الهيكل المقترح والذي يبدأ بترميز النموذج ثلاثي الأبعاد وتحضير جداول البيانات ثم تحليل ودراسة البيانات وإيجاد العلاقات بينها لتعديلها وتجهيزها بطرق مختلفة للاسـتخدام المباشـر مع تقنيات تعلم الآلة. ثم يتم شرح كيفية تقسيم البيانات لبيانات تستخدم في تمرين الأنظمة وبيانات تستخدم في الاختبار قبل عرض طريقة تمرين الأنظمة المختلفة.

*الباب السادس: تحليل تعلم الآلة والنتائج*

في الباب الأخير يتم عرض وتحليل نتائج تعلم الأنظمة المختلفة ومقارنتها مع بعضها البعض. ثم يتم توضيح كيفية استخدام النتيجة للتنبؤ بالمعاملات المختلفة التي يتم إدخالها لنموذج الترميز للحصول على نموذج ثلاثي الأبعاد يمكن التعديل عليه من قبل المعماري بطريقة سريعة وسهلة. وينتهي الباب بمناقشة النتائج.

**النتائج والتوصيات**

بناءً على ترتيب الأجزاء والأبواب يتم عرض النتائج المتعلقة بالدراسة ابتداءً من بعض الآراء التحليلية لاستخدام الذكاء الاصطناعي وتعلم الآلة في عملية التصميم المعماري وصولاً للتجربة التي تم عملها وتوضيحها في الباب الأخير. وينتهي البحث ببعض التوصيات لأفكار بحثية مستقبلية بإمكانها تطوير النتيجة والإضافة إليها بشكل كبير.