T.C
İSTANBUL KÜLTÜR UNİVERSİTY
INSTITUTE OF GRADUATE STUDIES

DATA-DRIVEN-BASED MECHANICAL ANALYSIS AND DESIGN PROCESS USING SUPERVISED MACHINE LEARNING: STEEL CANTILEVER STUDY CASE

MASTER'S THESIS
ABDULLATIF MAHMOUD
2100000951

Department: Civil Engineering
Program: Structural Engineering

Thesis Advisor: Assist. Prof. Dr. Erdal COŞKUN

July 2025

T.C
İSTANBUL KÜLTÜR UNİVERSİTY
INSTITUTE OF GRADUATE STUDIES

DATA-DRIVEN-BASED MECHANICAL ANALYSIS AND DESIGN PROCESS USING SUPERVISED MACHINE LEARNING: STEEL CANTILEVER STUDY CASE

MASTER'S THESIS

Abdullatif Mahmoud
(2100000951)

Department: Civil Engineering
Program: Structural Engineering

Supervisor: Assist. prof. Dr. Erdal COŞKUN
Jury members: prof. Dr. Necmettin Gündüz
Dr. Edip Seçkin

July 2025

DECLARATION

As the sole author of this thesis, titled *"Data-Driven-Based Mechanical Analysis and Desgin Process Using Supervised Machine Learning: Steel Cantilever Study Case"*. I, Abdullatif Mahmoud, affirm that the work presented herein is entirely my original creation, completed in fulfillment of the requirements for the master's degree in the Faculty of Civil Engineering. I further certify that no part of this thesis, either in whole or in part, has been submitted or presented for consideration at any other university or academic institution as part of any degree or research paper. (03/07/2025)

Abdullatif Mahmoud

DEDICATION

I am deeply grateful to Almighty for granting me the strength, perseverance, and guidance to complete this work independently. It is my earnest hope that this effort will contribute meaningfully to the community.

I extend my heartfelt thanks to everyone who has supported this journey, whether by offering advice, sharing insights, or simply encouraging me along the way. In particular, I wish to express my profound gratitude to my thesis advisor, Asst. Prof. Erdal Coşkun, whose invaluable guidance and dedication were instrumental throughout the writing process.

I also appreciate the encouragement and support of my friends, colleagues, and acquaintances, especially my colleague who generously lent their assistance during critical moments.

Above all, I dedicate this work to my beloved mother, whose unwavering support, prayers, and encouragement have been a constant source of inspiration throughout my life.

TABLE OF CONTENTS

ABBREVIATION

ML	: Machine Learning
AI	: Artificial Intelligence
DL	: Deep Learning
LR	: Linear Regression
DTR	: Decision Tree Regressor
RFR	: Random Forest Regressor
API	: Application Programmer Integration
MSE	: Mean Squared Error
R^2	: R- Squared Score
SHM	: Structural Health Monitoring
SVM	: Support Vector Machine
PCA	: Principal Component Analysis
CNNs	: Convolutional Neural Networks
GAs	: Genetic Algorithms
ANN	: Artificial Neural Networks
IoT	: Internet of Things
FRP	: Fiber Reinforced Polymer
RC	: Reinforced Concrete
K-NN	: K-nearest neighbor
GEP	: Gene Expression Programming
HPC	: High-performance concrete
AE	: Acoustic Emission
ASR	: alkali-silica reaction
SCC	: Stress corrosion cracking
KEDL	: Knowledge-enhanced deep learning
LSTM	: Long short-term memory
SDOF	: Single-degree-of-freedom
MDOF	: Multi-degree-of-freedom
CP	: Collapse potential
RSM	: Response surface methodology
ANFIS	: Adaptive neuro-fuzzy inference system
IDA	: Incremental dynamic analysis
MAE	: Mean absolute error
UAVs	: Unmanned aerial vehicles
BIM	: Building Information Modeling

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

δ	: Vertical displacement
$\delta_{allowable}$	: Allowable deflection according to acceptance criteria.
E	: Modulus of Elasticity
I	: Moment of Inertia
L	: Length
P	: Load
U1	: Deflection in the x-axis due to applied load
U2	: Deflection in the y-axis due to applied load
U3	: Deflection in the z-axis due to applied load
R1	: Rotating in the x-direction due to applied load
R2	: Rotating in the y-direction due to applied load
R3	: Rotating in the z-direction due to applied load

Denetimli Makine Öğrenmesi Kullanılarak Veri Odaklı Mekanik Analiz ve Tasarım Süreci: Çelik Konsol Kiriş Çalışması

ÖZET

Yapısal mühendislikte Yapay Zeka (AI) ve Makine Öğrenmesi (ML) kullanımı, tahmin doğruluğunun artırılması, zaman ve maliyet verimliliği sağlanması ve tasarım süreçlerinin optimize edilmesi gibi önemli avantajlar sunmaktadır. Veri odaklı içgörüler sağlayarak, bu teknolojiler daha gelişmiş yük tahmini, gerçek zamanlı yapısal sağlık izleme ve sürdürülebilir, kaynak verimli tasarımların gerçekleştirilmesine imkân tanır. Ancak, bu avantajların elde edilmesi; yüksek kaliteli ve güvenilir verilerin sağlanması ve karmaşık modellerin hesaplama taleplerinin yönetilmesi gibi bazı zorlukları da beraberinde getirir. Ayrıca, yapay zekanın geleneksel mühendislik iş akışlarına entegrasyonu, model yorumlanabilirliği, farklı yapısal türlere genelleme yapabilme ve mühendislerin yeni teknolojiler konusunda eğitilmesi gibi sorunların aşılmasını gerektirir. Bu engellere rağmen, yapay zekâdaki ilerlemeler, yapısal analizlerde entegrasyonunu giderek daha uygulanabilir hâle getirmekte, böylece daha güvenli ve yenilikçi tasarımların önünü açmaktadır.

Bu tezin temel amacı, makine öğrenmesi tekniklerinin – özellikle Doğrusal Regresyon (Linear Regression), Karar Ağacı Regresörü (Decision Tree Regressor), Rastgele Orman Regresörü (Random Forest Regressor) ve Karar Ağacı Sınıflayıcısı (Decision Tree Classifier) – çelik konsol kirişlerde I-kesit tasarımı ve düşey yer değiştirme gibi anahtar yapısal parametreleri tahmin etmedeki etkinliğini değerlendirmektir. I-kesit tasarımının (W-profili) elastisite modülü, kiriş uzunluğu ve yük gibi giriş parametrelerine dayanarak, düşey yer değiştirmenin ise kiriş uzunluğu, yük, elastisite modülü ve W-profili gibi parametreler üzerinden tahmin edilmesi hedeflenmiştir. Bu modellerin tahmin doğruluğu ve performansı değerlendirilerek, makine öğrenmesinin yapısal tasarım süreçlerine entegrasyonunun uygulanabilirliği ve avantajları belirlenmeye çalışılmıştır. Nihai amaç, yapısal mühendislikte verimliliği, doğruluğu ve veri odaklı karar alma süreçlerini geliştirmektir.

Çalışma üç ana adımdan oluşmaktadır. İlk adımda, bağımsız parametreler (elastisite modülü, uzunluk, yük) rastgele seçilerek ETABS yazılımına girilmiş, I-kesit tasarlanmış ve uygun W-profili belirlenmiştir. W-profili belirlendikten sonra, ETABS'ten düşey yer değiştirme (U3) verileri alınmıştır. İkinci adımda, oluşturulan bu veri seti kullanılarak makine öğrenmesi modelleri – özellikle Doğrusal Regresyon, Karar Ağacı Regresörü ve Rastgele Orman Regresörü – eğitilmiş ve düşey yer değiştirme tahmini yapılmıştır; Karar Ağacı Sınıflayıcısı ise I-kesit tasarımını tahmin etmek için kullanılmıştır. Son adımda ise, eğitilen modeller Ortalama Kare Hata (MSE) ve R-Kare Skoru (R^2) gibi performans ölçütleri kullanılarak değerlendirilmiş ve bu makine öğrenmesi yöntemlerinin yapısal analiz ve tasarım uygulamalarındaki potansiyelleri analiz edilmiştir.

Tezin hedeflerini doğrulamak amacıyla bir vaka çalışması yapılmıştır. Bu çalışma iki temel adımdan oluşmaktadır: İlk olarak, I-kesit seçimini tahmin etmek için Karar Ağacı Sınıflayıcısı kullanılmış, ardından düşey yer değiştirme (U3) tahmini için regresyon analizi uygulanmıştır. Bu yaklaşım, makine öğrenmesi modellerinin performans kriterlerine dayalı olarak optimize tasarım parametreleri önermedeki etkinliğini değerlendirmiş ve yapısal verimlilik ile doğruluğu artırmadaki potansiyelini ortaya koymuştur.

Veri seti toplam 2.601 veri çiftinden oluşmaktadır. Model performansını değerlendirmek için veri seti eğitim ve test setlerine bölünmüş, %20'lik test oranı kullanılmıştır. Bu araştırma, inşaat mühendisliğinde yapısal analiz yöntemlerinin geliştirilmesine önemli bir katkı sunmaktadır. Özellikle, makine öğrenmesi modelleri, konsol kirişlerin farklı koşullar altında gösterdiği davranışları doğru şekilde tahmin etmede umut verici performans sergileyerek değerli içgörüler sağlamıştır.

Anahtar Kelimeler: *Yapısal analizde makine öğrenmesi, yapısal tasarımda makine öğrenmesi, konsol kirişler, çelik, tahmine dayalı modelleme, düşey yer değiştirme, tasarım optimizasyonu.*

Data-Driven-Based Mechanical Analysis and Desgin Process Using Supervised Machine Learning: Steel Cantilever Study Case

ABSTRACT

The use of AI and ML in structural engineering offers significant benefits, including improved prediction accuracy, time and cost efficiency, and optimized design processes. By enabling data-driven insights, these technologies allow for enhanced load prediction, real-time structural health monitoring, and sustainable, resource-efficient designs. However, achieving these benefits also presents challenges, such as ensuring high-quality, reliable data and managing the computational demands of complex models. Additionally, integrating AI into traditional engineering workflows requires overcoming issues related to model interpretability, generalization to diverse structural types, and training engineers in new technologies. Despite these obstacles, advancements in AI continue to make its integration in structural analysis more feasible, paving the way for safer and more innovative designs.

The primary aim of this thesis is to evaluate the effectiveness of machine learning techniques -specifically Linear Regression, Decision Tree Regressor, Random Forest Regressor, Decision Tree Classifier- in predicting key structural parameters, such as I-section design and vertical displacement of steel cantilever beams. Predicting I-section design (W-shape) based on input parameters like modulus of elasticity, length and load while predicting vertical displacement based on input parameters like beam length, load, modulus of elasticity, and W-shape. By assessing the predictive accuracy and performance of these models, this research seeks to determine the feasibility and benefits of integrating machine learning into structural design processes, ultimately aiming to enhance efficiency, accuracy, and data-driven decision-making in structural engineering.

The study consists of three main steps. First, the dataset was prepared which contained pairs of data, independent parameters (modulus of elasticity, length, load) were arbitrarily selected and input into ETABS to design the I-section and determine the appropriate W-shape. Once the W-shape was identified, the vertical displacement (U3)

was retrieved from ETABS. Second, this dataset is used to train machine learning models -specifically, Linear Regression, Decision Tree Regressor, and Random Forest Regressor- aimed at predicting vertical displacement while Decision Tree classifier used to predict I-section design. Lastly, the trained models are evaluated using Mean Squared Error (MSE) and R- Squared Score (R^2) to assess their predictive accuracy and effectiveness, providing insights into the applicability of these machine learning methods in structural analysis and design.

To validate the aims of this thesis, one case study was conducted. It consists of two main steps: first, the Decision Tree Classifier is applied to predict the I-section selection; second, regression analysis is used to predict vertical displacement (U3). This approach assessed the machine learning model's effectiveness in recommending optimized design parameters based on performance criteria, demonstrating their potential in enhancing structural efficiency and accuracy.

The dataset consists of 2601 rows of data pairs. The datasets were divided into training and testing sets to evaluate model performance. Testing size of 20% was utilized to train and assess model accuracy. This research contributes to the advancement of structural analysis methodologies in civil engineering. Notably, machine learning models exhibit promising performance in accurately predicting displacement, thereby offering insights into the behavior of cantilever beams under different conditions.

Keywords: *Machine learning in structural analysis, machine learning in structural design, cantilever beams, steel, predictive modeling, vertical displacement, design optimization*

1 INTRODUCTION

1.1 Background

Artificial Intelligence (AI) and Machine Learning (ML) are at the forefront of innovation in civil engineering, driving data-driven solutions across diverse subfields. AI refers to the development of systems capable of performing tasks that typically require human intelligence, while ML, as a subset of AI, enables machines to learn from data autonomously. These technologies are reshaping traditional engineering practices, offering enhanced accuracy, efficiency, and sustainability in problem-solving.

In water resources, ML models forecast groundwater levels, analyze water consumption patterns, and predict failures in water distribution systems. In geotechnical engineering, AI is applied to model soil behavior, assess slope stability, and optimize foundation designs. Construction engineering benefits from ML in project scheduling, cost estimation, and quality control by analyzing extensive datasets. Environmental engineering leverages AI to model air and water quality, manage waste systems, and evaluate environmental impacts. In transportation engineering, AI and ML enable traffic flow prediction, congestion management, and vehicle routing, contributing to more efficient and reliable infrastructure systems [1].

Structural engineering has particularly embraced AI and ML to enhance design accuracy and analysis efficiency. These technologies empower engineers to develop predictive models for structural behaviors, such as the shear capacity of reinforced concrete beams, the fundamental period of buildings, and the deflection of curved steel beams. By processing large datasets and using advanced algorithms, AI and ML deliver more precise results than traditional methods, improving structural safety, cost-efficiency, and sustainability [2]. Figure 1.1 illustrates the application of AI in a neural network with one input layer of two neurons, one hidden layer of four neurons, and

one output layer of three neurons, showcasing the complexity and capability of these systems.

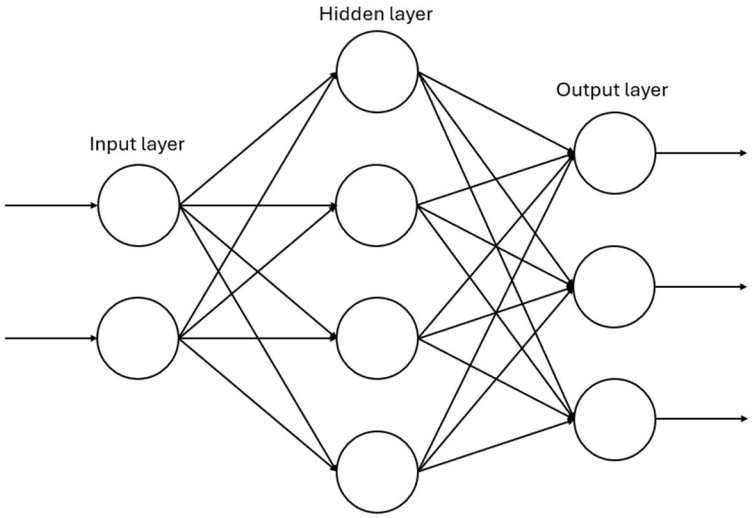


Figure 1.1: Typical AI neural networks.

Machine learning techniques have further revolutionized structural engineering in key areas like load prediction, structural health monitoring (SHM), and design optimization. ML models analyze historical and real-time data to predict structural loads, enhancing safety and durability. For instance, neural networks can predict wind loads on buildings using historical weather data, while Support Vector Machines (SVM) estimate traffic loads on bridges based on traffic patterns and vehicle weights. In SHM, ML processes large volumes of sensor data to detect early signs of structural damage or deterioration. Techniques like Principal Component Analysis (PCA) analyze vibration data from bridges to identify anomalies, while Convolutional Neural Networks (CNNs) detect surface cracks in buildings through image analysis [1].

Design optimization, another critical application, employs ML to maximize structural performance while minimizing costs. Genetic Algorithms (GAs) refine the design of structural components, and ML models suggest optimal material combinations for construction. These advancements in load prediction, SHM, and design optimization not only enhance the safety and reliability of structures but also promote innovative and sustainable engineering practices. Through the integration of AI and ML, structural engineering continues to evolve, addressing modern challenges with efficiency and precision [3].

1.2 Motivation

Finite Element Analysis (FEA) software, such as ETABS, has long been a cornerstone of structural engineering practice -providing accurate, reliable, and well-validated results. This study does not seek to replace these tools, but rather to complement and extend their capabilities through the integration of machine learning (ML) techniques.

Currently, FEA tools serve the profession well. However, in an era defined by rapid technological progress, the future of structural engineering will increasingly depend on adaptive, intelligent, and data-driven methods. The construction and infrastructure sectors are under constant pressure to innovate -demanding faster decision-making, smarter design processes, and more sustainable solutions. Machine learning offers a powerful opportunity to address these needs by uncovering patterns in structural behavior, automating routine tasks, and enhancing the efficiency of design workflows.

This research is motivated by the desire to explore how ML can augment traditional engineering tools, enabling predictive modeling and optimization in structural design. By leveraging high-quality simulation data from ETABS, this study aims to demonstrate how ML can deliver accurate predictions of structural responses, inform design selection, and potentially pave the way for more flexible, scalable, and intelligent engineering systems. Adopting such forward-thinking approaches ensures that the field remains at the forefront of innovation -prepared not just for today's demands, but for the uncertainties and opportunities of tomorrow.

Moreover, this study contributes to bridging the gap between theoretical innovation and practical application. By integrating machine learning models into familiar engineering environments like ETABS, it offers a pathway for gradual adoption rather than disruptive overhaul. This approach encourages engineers and researchers to engage with emerging technologies in a controlled, testable manner—grounded in real-world data and industry standards. As a result, the research not only advances academic understanding of ML in structural analysis but also provides actionable insights that can support future developments in smart infrastructure, automated design systems, and performance-based engineering.

1.3 Problem Statement

This study investigates the use of data-driven prediction methods, specifically Machine Learning (ML), in enhancing the structural analysis and design selection processes. As outlined in the abstract, data-driven approaches leverage historical (collected) data and advanced algorithms to predict structural behaviors and optimize designs. This research aims to address the following inquiries:

1) Investigate the ability of data-driven prediction methods and Machine Learning (ML) to predict key structural parameters of beams such as vertical displacement based on structure features.
2) Investigate ability of data-driven prediction methods and Machine Learning (ML) to predict I-section design of beams based on structure configuration.
3) Evaluate the performance and efficiency of these methods to predict structural analysis results (key structural parameters) of beams.
4) Evaluate the performance and efficiency of these methods to predict optimal structure design (I-section design) of beams.

This research is bounded by specific limitations, concentrating solely on the prediction of vertical displacement and I-section (W-shape) design for steel cantilever beams subjected to a single point load at the free end, utilizing machine learning techniques. The scope is restricted to ETABS software simulation results, and only cantilever configurations are considered. Figure 1.2 below illustrates a typical cantilever beam subjected to a point load at its free end, representing the structural system analyzed throughout this study.

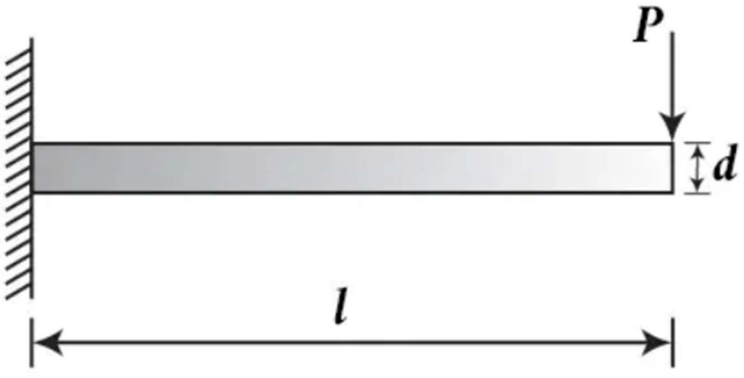


Figure 1.2: Typical cantilever beam subjected to point load P.

1.4 Objectives

This study focuses on the use of data-driven prediction methods and Machine Learning (ML) in the field of structural engineering and its potential to enhance design optimization. By leveraging large datasets and advanced algorithms, data-driven approaches can improve the accuracy and efficiency of structural design. The key objectives of this research are:

1) Model's performance or accuracy of ML methods in structural analysis problems.
2) Model's performance or accuracy of ML methods in design selection problems.

The primary goal is to investigate how data-driven prediction methods can be used to predict key structural parameters and support design selection. This study aims to assess the feasibility and advantages of using these methods in modern structural design and analysis.

1.5 Research Significance

This research holds significance as it explores the potential of machine learning models -Linear Regression, Decision Tree Regressor, and Random Forest Regressor- to accurately predict key structural behaviors such as vertical displacement and employs the Decision Tree Classifier to determine the I-section design of cantilever beams. By automating these predictions using parameters like beam length, applied load, and modulus of elasticity, the study seeks to enhance the efficiency, safety, and cost-effectiveness of structural design processes. It underscores the increasing impact of data-driven approaches in reshaping traditional engineering practices, providing a more efficient and innovative framework for structural analysis and optimization.

1.6 Scope and Aim of the Thesis

This research focuses on evaluating the effectiveness of three machine learning regression models -Linear Regression, Decision Tree Regressor, and Random Forest Regressor- in predicting the vertical displacement of steel cantilever beams. The predictions are based on key structural parameters, including beam length, applied point load, and modulus of elasticity. Additionally, the study investigates the effectiveness of the Decision Tree Classifier in predicting the selection of an

appropriate I-section (W-shape) based on parameters such as beam length and applied load.

The study explores a range of cantilever beam configurations with varying geometric properties, while keeping material characteristics -modulus of elasticity (E) and yield strength (Fy)- constant throughout the analysis. By examining the performance and accuracy of these machine learning models, the research aims to assess their feasibility for structural engineering applications. The findings are intended to contribute to the ongoing development of AI-driven approaches in structural design and analysis, offering insights into the potential of machine learning to enhance engineering workflows and optimize structural performance.

1.6 Layout of the Thesis

To accomplish the primary aim of this research, fulfill the stated objectives, and address the research questions, the thesis is organized into the following six chapters:

Chapter 1: Introduction

This chapter presents an overview of the research topic, defines the objectives, formulates the research questions, and outlines the significance and scope of the study.

Chapter 2: Literature Review

This chapter provides a comprehensive review of the historical development and application of machine learning techniques in structural engineering. It examines their relevance, effectiveness, and success rates, while comparing findings from similar studies. The chapter concludes by summarizing key insights and establishing the contextual foundation for the present research.

Chapter 3: Methodology

This chapter details the problem definition, focusing on variables such as beam length, load, modulus of elasticity. It describes data collection, preprocessing, and splitting dataset into training and testing sets. Machine learning models -Linear Regression, Decision Tree Regressor, and Random Forest Regressor- are developed and trained on the processed data to predict vertical displacement. Additionally, a Decision Tree Classifier is implemented to predict the appropriate W-shape I-section. The performance of all models is evaluated using Mean Squared Error (MSE) and R-squared (R^2) metrics to assess their predictive accuracy and effectiveness.

Chapter 4: Implementation and Results

The methodology is implemented using a dataset of 2601 data rows and 20% testing/training splits. It consists of two main steps: first, the Decision Tree Classifier is applied to predict the I-section selection; second, regression analysis is used to predict vertical displacement (U3).

Chapter 5: Discussion and Conclusion

This chapter interprets the results, providing an analysis of model performance, supported by sample calculations and insights into the implications of the findings. A summary of the research outcomes is presented, including key findings and recommendations for practical implementation. Suggestions for future research directions are also provided.

2 LITERATURE REVIEW

The integration of Artificial Intelligence (AI) and Machine Learning (ML) into structural engineering has become a focal point of research, offering transformative potential for traditional practices. These advanced technologies enable improved prediction accuracy, efficiency, and optimization of design processes -critical components in addressing the increasing complexity and demands of modern structural engineering. This literature review explores the evolution of machine learning techniques within the field, emphasizing their significance and supported by publications related to the subject. Additionally, it provides a comprehensive summary and comparison of relevant studies, shedding light on the progress and challenges in utilizing AI and ML for structural analysis and design [4].

2.1 History of ML Techniques in Structural Engineering

Structural engineering has increasingly adopted Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) to tackle complex challenges. These technologies offer data-driven solutions that enhance prediction accuracy and design efficiency. Their integration has evolved significantly, transforming traditional workflows. The following sections explore this progression, from early developments to modern applications. They also highlight the transformative potential of AI-driven methods. Despite progress, standardized approaches and reliable metrics are still needed for broader implementation.

2.1.1 Evolution of Machine Learning Techniques within the Field

The integration of Machine Learning (ML), Artificial Intelligence (AI), and Deep Learning (DL) into structural engineering has evolved over several decades, marked by gradual adoption and significant advancements. This progression can be categorized into distinct phases, from early logic-based systems in the 1980s to advanced AI models today, significantly transforming the approach to structural engineering challenges.

In the 1980s, the application of AI in structural engineering began with logic-based systems primarily designed for exploratory design tasks. These early systems were constrained by limited computational power and data availability, which restricted their capabilities. By the 1990s, advancements in computational power and algorithmic sophistication enabled the broader application of AI techniques. During this period, AI was increasingly utilized for optimization problems and preliminary design tasks, offering more efficient approaches to solving structural engineering challenges.

The 2000s saw a pivotal shift with the introduction of ML algorithms, such as Support Vector Machines (SVM) and Artificial Neural Networks (ANN). These algorithms facilitated applications in predicting material properties, structural health monitoring, and damage detection, marking a significant milestone in the use of data-driven methods in structural engineering.

In the 2010s, the advent of big data and the Internet of Things (IoT) brought transformative changes to structural health monitoring systems. Real-time data collection and processing became integral, while Convolutional Neural Networks (CNNs) emerged as powerful tools for image-based damage detection, improving the accuracy and reliability of structural assessments.

Most recently, in the 2020s, the use of AI, ML, and DL has expanded to encompass various structural engineering subfields, including earthquake engineering, wind engineering, and fire engineering. The emphasis has shifted towards developing scalable and robust AI models that can address the complexities of modern structural engineering projects, paving the way for innovative and efficient solutions [5][6].

2.1.2 Publications related to the Subject

The integration of Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) into structural engineering has the potential to revolutionize the field with innovative, cost-effective solutions that surpass traditional methods. While recent advancements have demonstrated improved predictions, optimizations, and assessments, there remains a need for systematic methodologies, reliable performance metrics, and standardized dataset approaches to fully harness these technologies.

(Teymori Gharah Tapeh and Naser, 2023) address this gap through a scientometric analysis of over 4,000 scholarly works, revealing a significant rise in AI-related publications in structural engineering over the past decade. Their review identifies best

practices in methodology, performance evaluation, and data management, offering a roadmap for future research and application.

This growing body of work emphasizes the importance of ongoing study, education, and collaboration to overcome challenges and unlock the full transformative potential of AI, ML, and DL in structural engineering. Figure 2.1 illustrates the trend of publications adopting AI in the field.

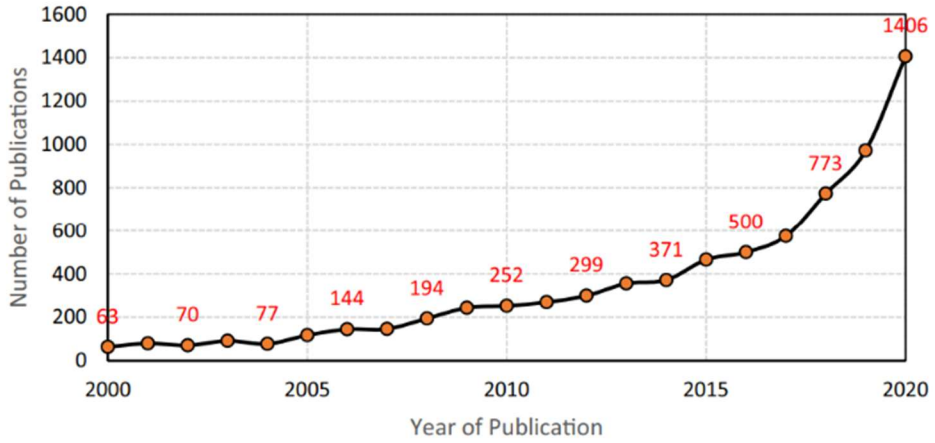


Figure 2.1: Publications adopting AI derivatives in structural engineering [5].

Figure 2.2 illustrates a monograph showcasing the annual publication trends from 2011 to 2020. The articles are organized into categories based on specific keywords, including "artificial intelligence and structural engineering," "machine learning and structural design," and "deep learning and earthquake engineering," among others.

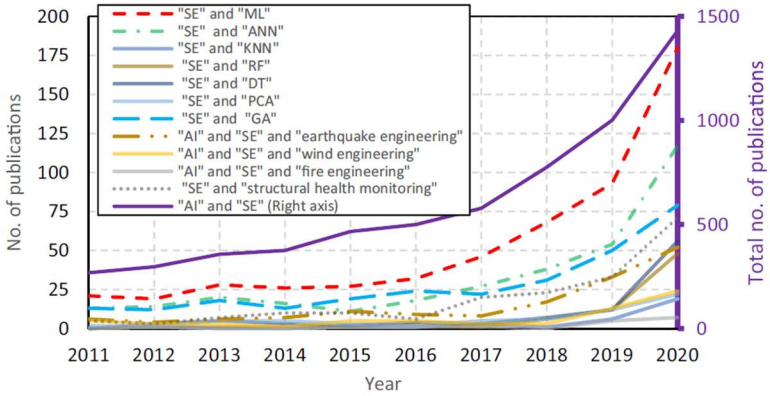


Figure 2.2: The number of publications per year between 2011 and 2020 [5].

Figure 2.3 presents a bar chart highlighting the top twenty journals with the highest number of published articles on AI, ML, and DL in structural engineering and earthquake-related studies. Leading the list is Computer-Aided Civil and Infrastructure Engineering, which accounts for approximately 8% of the total publications.

Construction and Building Materials follows as the second-ranking journal, contributing just under 4% of the publications.

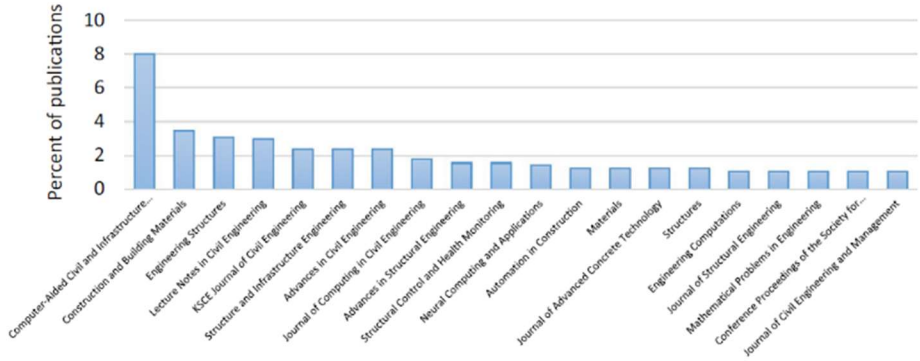


Figure 2.3: Frequently publishing journals [5].

2.2 Applications of ML in Structural Engineering

Machine Learning (ML) has demonstrated transformative potential across various aspects of structural engineering by enhancing analysis, monitoring, and optimization processes. Its applications can be broadly categorized into structural health monitoring, material behavior prediction, load forecasting, and design optimization.

2.2.1 Structural Health Monitoring (SHM)

Machine Learning (ML), Artificial Intelligence (AI), and Deep Learning (DL) are transforming Structural Health Monitoring (SHM) by improving the accuracy, efficiency, and automation of damage detection and evaluation processes [7].

(Zhang et al., 2022) demonstrated the use of AI in SHM through convolutional neural networks (CNNs) for fault detection in marine diesel engines. By analyzing vibration signals, CNNs effectively distinguish between normal and faulty conditions, enhancing maintenance reliability.

(Soltangharaei, Ai, and Ziehl, 2023) employed Acoustic Emission (AE) technology with machine learning techniques like k-means clustering, Principal Component Analysis (PCA), and deep learning models to monitor concrete and steel structures. This approach enabled the early detection of damage mechanisms such as stress corrosion cracking and alkali-silica reaction, improving infrastructure safety.

(Azimi, Eslamlou, and Pekcan, 2020) reviewed advancements in SHM with a focus on deep learning methods, including neural networks and transfer learning. Their study emphasized integrating technologies such as UAVs, sensors, and cloud computing to enhance SHM capabilities. They highlighted the challenges of data quality, sensor

deployment, and computational demands while showcasing DL's potential in vibration-based and vision-based monitoring.

(Andrushia et al., 2023) developed a CNN model to detect and quantify thermal cracks in fire-damaged concrete. By analyzing high-resolution images of heated concrete specimens, their model accurately correlated crack patterns with residual compressive strength, demonstrating a robust tool for assessing fire-induced damage.

In conclusion, integrating ML, AI, and DL into SHM systems enhances the ability to monitor and maintain structural integrity, offering significant benefits in terms of safety, efficiency, and cost reduction.

2.2.2 Prediction of Material Behavior

Artificial Intelligence (AI) and Machine Learning (ML) are transforming material innovation by dramatically improving the prediction, analysis, and understanding of complex material behaviors, leading to faster discoveries and more efficient material designs. For instance:

(Wang et al., 2024) proposed an innovative ML framework integrating domain knowledge to predict the bearing capacity of concrete-filled steel tube (CFST) columns. Utilizing a database of 2621 experimental data points, they developed a Domain Knowledge Enhanced Neural Network (DKNN) model with advanced feature engineering techniques, including Pearson correlation, XGBoost, and Random Tree algorithms. The model demonstrated a substantial improvement in prediction accuracy, reducing the Mean Absolute Percentage Error (MAPE) by over 50% compared to existing models, and maintained robust performance even in noisy environments.

(Tarawneh and Saleh, 2023) demonstrated the use of finite element analysis (FEA) combined with artificial neural networks (ANN) to predict the impact resistance of fiber-reinforced polymer (FRP)-reinforced concrete beams, achieving a maximum variation of just 6.6% from FEA results. Similarly,

(Mansouri, Tezcan, and Awoyera, 2023) employed a radial basis function (RBF) model integrated with metaheuristic algorithms to predict the compressive strength of high-performance concrete (HPC). The Sine Cosine Algorithm (SCA) combined with the RBF model yielded the most accurate results, improving prediction precision and reducing errors compared to traditional methods.

(JRE Editorial Office et al., 2024) highlighted the importance of funding in advancing research and innovation is underscored in this study. Government grants, such as those from the U.S. Small Business Administration (SBA), and institutional support from entities like the National Science Foundation (NSF), provide critical resources for groundbreaking research and development, driving progress across technology, medicine, and environmental science.

In conclusion, the application of AI and ML in material science accelerates the discovery of new materials while enhancing the accuracy and reliability of material behavior predictions. These advancements offer transformative and efficient engineering solutions that push the boundaries of traditional practices.

2.2.3 Design Optimization

Machine learning (ML) is transforming design optimization in structural engineering by enabling efficient exploration of diverse design possibilities. Techniques like Genetic Algorithms (GAs) and Decision Trees optimize structural components, minimizing material usage while preserving strength and functionality. Additionally, ML evaluates multiple design parameters simultaneously, aiding in the selection of optimal configurations.

(Ampanavos et al., 2021) introduced ApproxiFramer, an ML-based tool to assist architects in the conceptual design phase. By generating structural layouts from building plan sketches in real time using a Convolutional Neural Network (CNN) trained on synthetic datasets, the system achieves an average error of just 2.2% in predicting column positions. This tool integrates structural engineering knowledge into early-stage design, streamlining updates and enabling more efficient architectural decision-making.

In seismic design, (Wasse et al., 2024) highlighted the performance of special concentrically braced frames (SCBFs) for various building types. While SCBFs are effective in providing seismic resistance, the study identifies research gaps in assessing SCBFs in complex industrial structures. The authors advocate integrating AI techniques to enhance SCBF performance and facilitate post-earthquake damage detection, thus improving the resilience of industrial buildings.

(Wang et al., 2024) developed hybrid ML models combining artificial neural networks (ANN) with optimization algorithms like genetic algorithms (GA) and particle swarm

optimization (PSO) to predict the bond strength of concrete-encased steel (CES) structures. Using a dataset of 191 records from push-out tests, the PSO-ANN model demonstrated superior prediction accuracy and convergence speed compared to conventional methods. Sensitivity analysis revealed the relative concrete cover's significant impact on bond strength, and the study introduced a user-friendly design tool based on the hybrid model.

(Fu et al., 2023) presented FrameGAN, a novel method leveraging dual generative adversarial networks (GANs) to automate steel frame-brace layout design. Tested against mainstream GAN models like pix2pix and pix2pixHD, FrameGAN outperformed these methods in generating accurate and efficient layouts. By automating the design process, FrameGAN reduces manual effort and improves design quality, streamlining structural engineering workflows.

In summary, the integration of ML into design optimization enables more efficient, accurate, and innovative solutions, driving advancements in architectural planning, seismic design, material performance prediction, and automated layout generation.

2.2.4 Automation in Structural Analysis

Machine learning (ML) is increasingly utilized to automate repetitive tasks in structural analysis, such as predicting structural responses to various loading scenarios. This automation allows engineers to focus on more complex aspects of structural design and analysis, enhancing efficiency and precision.

(Koodiani et al., 2023) demonstrated the efficacy of ML models in improving the accuracy of nonlinear modeling parameters for reinforced concrete (RC) columns under seismic loads. By leveraging ML regression models and neural networks, the study achieved more precise parameter estimates than traditional methods. Notably, Regularized Linear Regression and Polynomial Regression proved effective for specific parameters, while a multi-class classification tool enhanced the accuracy of predicting column failure modes. This advancement automates critical structural analysis tasks, making them more efficient and reliable.

(Le et al., 2023) highlighted the potential of ML models, specifically Random Tree (RT) and Artificial Neural Networks (ANN), in predicting the load-deflection behavior of composite concrete bridges. Among these, the RT model outperformed ANN, demonstrating higher prediction accuracy and reliability. The study underscores the

utility of RT in bridge health monitoring and management, providing precise load-deflection predictions that lead to significant time and cost savings.

(Van der Westhuizen et al., 2023) developed an Artificial Neural Network (ANN) model to predict the fundamental period of steel structures, addressing the limitations of traditional methods, especially in seismic regions with soil-structure interaction (SSI). Using an extensive dataset, the ANN model achieved a 99.9% correlation and a mean absolute percentage error (MAPE) of 0.7%, offering a highly accurate tool for seismic design. The study also introduced the Noesys_AutoML software for future enhancements and proposed exploring other ML algorithms to further improve predictive performance.

(Pham et al., 2020) analyzed an experimental dataset using various single and ensemble ML models. The study found that ensemble models like bagging Linear Regression (LR) and stacking models combining Multilayer Perceptron (MLP), Support Vector Regression (SMOreg), and LR delivered superior prediction accuracy. These models achieved high correlation coefficients ($R > 0.97$) and low mean absolute errors (MAE < 5 mm), demonstrating their effectiveness in forecasting long-term deflections. This contributes to better serviceability and safety evaluations of RC structures.

In summary, ML models enhance structural analysis by automating tasks, improving accuracy, and providing cost-effective solutions for complex engineering challenges. Their applications span seismic design, bridge health monitoring, and long-term deflection predictions, underscoring their transformative potential in structural engineering.

2.2.5 Evaluating the Sismic Performance of Structures

Machine learning (ML) techniques have emerged as powerful tools for evaluating the seismic performance of structures, such as buildings and bridges. By analyzing extensive datasets from past earthquakes, ML models can predict structural responses to seismic forces and recommend retrofitting strategies to enhance resilience.

(Wang, 2023) introduced the Knowledge-Enhanced Deep Learning (KEDL) methodology, which integrates prior knowledge, including physics-based and empirical formulas, into the loss function during training. This innovative approach reduces the need for large training datasets and enhances robustness to noise. KEDL

employs recurrent neural networks (RNNs) and long short-term memory (LSTM) neurons to model the excitation-response relationship of structures, while wavelet-domain projection simplifies the input-output dynamics, improving training efficiency. The methodology has demonstrated high accuracy, data efficiency, and generalization capability in single-degree-of-freedom (SDOF) and multi-degree-of-freedom (MDOF) systems.

In conclusion, ML models like KEDL provide accurate and efficient predictions of structural responses under seismic forces, facilitating safety analysis and predictive maintenance. These advancements contribute to the development of more resilient infrastructure systems.

2.3 Similar studies: Data-Driven Machine Learning Prediction of key Structural Parameters based on Structure Configuration

Machine learning (ML) techniques are proving to be transformative in the prediction of deflection and related parameters for various structural elements, enabling engineers to achieve higher accuracy and efficiency in their designs. These predictions leverage data on structural configurations, geometries, material properties, and loading conditions to deliver precise and reliable outcomes.

(Mustafa et al., 2024) showcased the application of Polynomial Regression (PR) in accurately predicting the central deflection and slenderness limits of simply supported concrete beams. Using a dataset of 400 samples and five input parameters (beam width, depth, length, load, and compressive strength), the PR model achieved near-perfect accuracy with R^2 values of 0.999 and 1.000 during training and testing. This study highlights the reliability of ML models for optimizing the lateral stability of concrete beams.

(Jin et al., 2022) explored ML models trained on finite element simulation data to predict the deflection and shape of 2D cantilever beams. The models achieved high accuracy, with a maximum error of 3.8%, and significantly reduced computation time compared to traditional methods. This research underscores the potential of ML in enhancing the efficiency of compliant mechanism design.

(Lai et al., 2023) utilized seven ML regression algorithms, including Support Vector Regression (SVR) and Random Forest Regressor (RFR), to predict the maximum displacements of reinforced concrete (RC) beams under impact loads. The study

achieved approximately 90% validation accuracy, with impact velocity and impactor mass identified as the most influential factors using the SHAP algorithm. These findings demonstrate the effectiveness of ML in modeling complex dynamic behaviors.

(Nguyen et al., 2023) proposed the WFR-FBI-LSSVR model for predicting long-term deflection in RC beams, integrating wrapper-based feature refinement (WFR) and forensic-based investigation (FBI) algorithms with least squares support vector regression (LSSVR). With an RMSE of 7.86 mm and an R^2 of 0.908, the model demonstrated high accuracy, offering a robust tool for early predictions of long-term deflections based on beam geometry, load conditions, material properties, and environmental factors.

(Ababu et al. 2023) focused on predicting the deflection of curved steel I-beams using deep artificial neural networks (ANNs). The ANN model outperformed traditional methods, achieving a mean absolute error of 6.4 mm for deflections and 30.43 kN for failure loads. This approach provides a more precise alternative for designing curved steel structures.

(Sethi, 2023) combined finite element analysis (FEA) with deep learning models, such as Feed Forward Neural Networks and Multi-Layer Perceptrons (MLP), to predict small deflections in steel cantilever beams. With percentage errors ranging from 0.5% to 10%, these models significantly reduced computation time compared to FEA alone, demonstrating the efficiency of integrating simulation-based data with ML.

In summary, ML models have become invaluable tools for accurately predicting deflection and other critical parameters in structural elements. They not only improve computational efficiency but also provide reliable and cost-effective solutions for complex engineering challenges across various structural applications.

2.4 Data-driven prediction of vertical displacement for steel cantilever beams based on structure configuration

Samarth Sethi's 2023 study, *"Optimizing the Prediction of Small Deflections for Steel Cantilever Beams with Finite Element Analysis and Deep Learning,"* provides a compelling example of adopting Deep Learning (DL) to enhance deflection prediction accuracy leveraging Finite Element Analysis (FEA) simulated data . This work specifically focused on steel cantilever beams, employing machine learning techniques

to predict small deflections based on beam geometry, material properties, and loading conditions.

The methodology involved the use of Artificial Neural Networks (ANNs), including MATLAB's Feed Forward Neural Network and a Multi-Layer Perceptron (MLP). These models were trained on data pairs derived from Autodesk Fusion's Static Stress simulations, which included parameters such as beam length, depth, width, modulus of elasticity, and load conditions.

The study demonstrated that the neural network models could achieve deflection predictions with a percent error ranging from 0.5% to 10%. This performance highlighted the potential of neural networks to serve as a reliable alternative to traditional analytical approaches like the Euler-Bernoulli beam theory. By integrating FEA-generated data with DL, this hybrid approach not only improved prediction accuracy but also significantly reduced computational complexity. This makes it a valuable tool for structural engineers, particularly in scenarios where efficiency and precision are paramount.

Sethi's research aligns closely with the objectives of the current study, particularly in its use of machine learning for structural parameter prediction. However, while Sethi's work focused on cantilever beams and employed FEA simulations for data generation, this study extends the scope by integrating both Euler-Bernoulli beam theory and Finite Element Analysis (FEA) simulated data. Additionally, the present research explores a broader range of structural elements, including I-section design optimization, with a specific focus on advanced machine learning models such as Decision Tree Regressor, Random Forest Regressor, and Random Forest Classifier. By incorporating ETABS, a more advanced FEM software compared to Autodesk Fusion used in Sethi's study, and applying innovative feature engineering, this research further improves the accuracy of deflection predictions and design optimizations. Building upon Sethi's foundation, this study aims to enhance the applicability of machine learning techniques in structural analysis and design optimization.

2.5 The Edge of this Research

This research leverages ETABS-simulated data to train Machine Learning (ML) models. A Decision Tree Classifier was employed to optimize I-section design selection. Additionally, Linear Regression, Decision Tree Regressor, and Random

Forest Regressor were used to predict vertical deflections in steel cantilever beams. The study achieved an I-section design prediction accuracy of 94.6% and a deflection prediction accuracy of up to 99.9%. These findings highlight a promising alternative to traditional methods, such as Finite Element software (e.g., ETABS) or analytical approaches like Euler-Bernoulli beam theory and allowable strength criteria, offering enhanced accuracy and reduced computational complexity for structural engineers.

The research employed a multi-step approach: data was collected on cantilever beam configurations (modulus of elasticity, W-shape, length, and load) and corresponding deflections. ML models were applied to predict deflections and I-section designs, with performance evaluated using a dataset of 2,601 rows, split into an 80/20 training and testing proportion.

One case study was conducted using ETABS-simulated data to examine non-linear structural behavior and design optimization. It consisted of two main steps: first, the Decision Tree Classifier was applied to predict I-section selection; second, regression analysis was used to predict vertical displacement (U3).

The findings highlighted that the Decision Tree Classifier was highly effective in predicting I-section design. Moreover, decision tree-based models demonstrated superior accuracy in deflection prediction compared to linear regression. The study stands out for its innovative use of Python to generate diverse structural configurations and its integration with ETABS, a more advanced FEM software than Autodesk Fusion, which was used in similar studies.

Overall, the research demonstrates the potential of ML models to enhance structural analysis and design efficiency, providing valuable tools for structural engineers seeking to improve prediction accuracy and optimize designs.

2.6 Supervised Machine Learning

It is a type of machine learning where an algorithm is trained on a labeled dataset. This means that the training data includes both the input data and the corresponding correct output. The goal is for the algorithm to learn mapping from inputs to outputs so that it can make accurate predictions on new, unseen data [30]. Figure 2.4 illustration shows the key points of supervised ML:

1) Labeled Data: The training dataset contains input-output pairs. For example, in a spam detection system, emails (inputs) are labeled as "spam" or "not spam" (outputs).
2) Training Process: The algorithm learns by comparing its predictions to the actual outputs and adjusting to minimize errors.
3) Applications: Common applications include classification tasks (e.g., image recognition, spam detection) and regression tasks (e.g., predicting house prices).

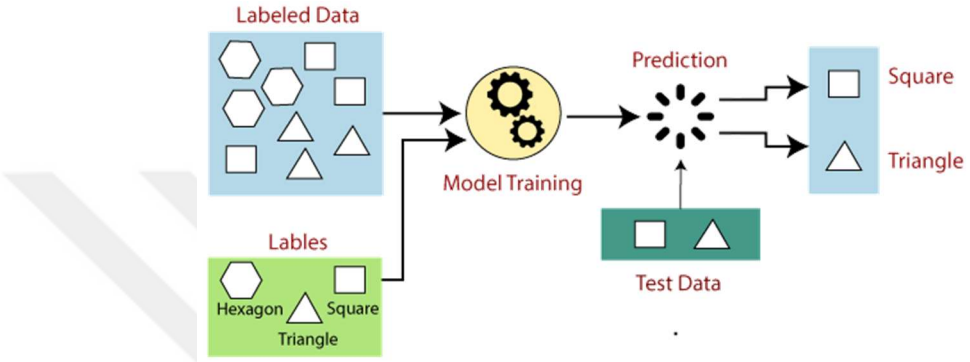


Figure 2.4: The illustration of supervised ML Model.

In contrast to unsupervised machine learning (ML), where the algorithm trains on unlabeled data and seeks to discover patterns and relationships without specific guidance on the desired output , supervised ML relies on labeled datasets. A common task in unsupervised learning is clustering, which involves grouping similar data points, as illustrated in Figure 2.5. An example of clustering is customer segmentation in marketing. Additionally, Figure 2.6 depicts Reinforcement Learning, a distinct type of machine learning where an agent learns to make decisions by interacting with an environment. The agent's goal is to maximize cumulative rewards by taking actions that lead to optimal outcomes over time [31 , 32].

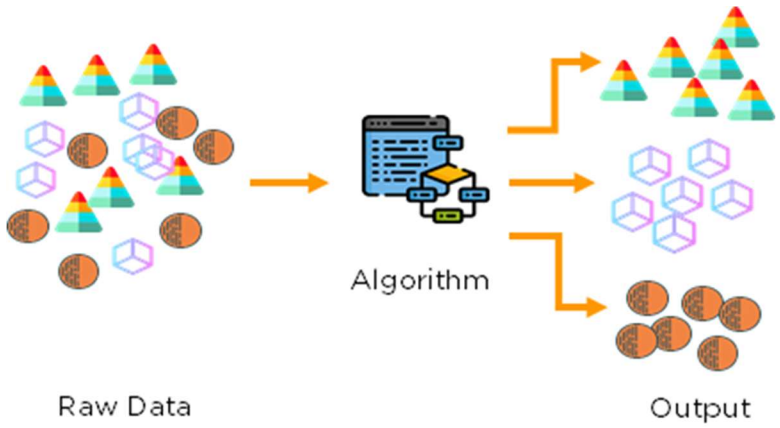


Figure 2.5: The illustration of unsupervised ML Model.

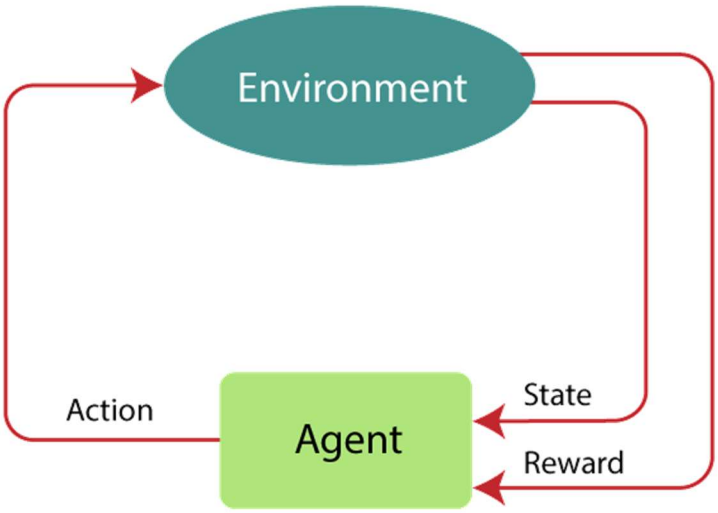


Figure 2.6: The illustration of Reinforcement Learning.

Supervised learning algorithms are essential tools in machine learning, designed to learn from labeled data to make predictions or decisions. These algorithms can be broadly categorized into two types: regression and classification, each suited to specific tasks [33, 34]. Regression algorithms are employed to predict continuous outcomes by modeling the relationship between dependent and independent variables. For example, regression can predict the vertical displacement of a cantilever beam based on features like modulus of elasticity, moment of inertia, length, and load. Conversely, classification algorithms are used to predict discrete outcomes by categorizing data into predefined classes. For instance, a classifier can determine the best I-section design for a cantilever beam based on features such as length and loads. Figure 2.7 below illustrates the different types of AI and ML, providing a comprehensive overview.

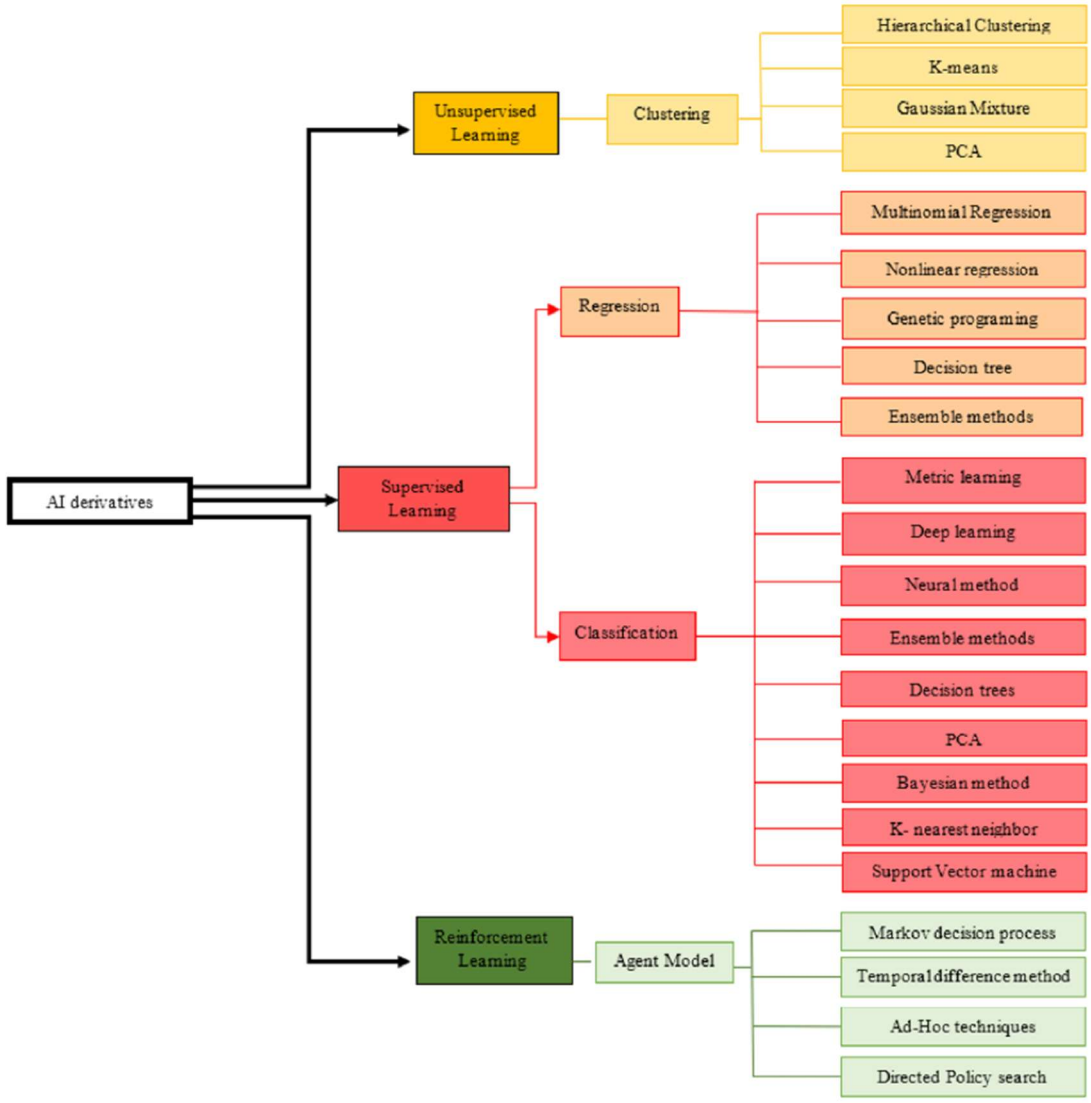


Figure 2.7: Learning Methods.

3 METHODOLOG

The thesis demonstrates its findings through a detailed case study built on a structured three-step methodology. In the initial phase, ETABS software is employed to simulate the behavior of cantilever beams under varying conditions. This involves extracting analytical results through the ETABS API for multiple configurations, including different beam lengths, applied loads, and material properties. The goal of this phase is to generate a comprehensive and diverse dataset that accurately captures the structural response of cantilever beams.

In the second phase, the study applies machine learning techniques to the collected data. Specifically, two models -Decision Tree Regressor and Decision Tree Classifier- are developed and trained. The Decision Tree Regressor is used to predict vertical displacement, a key performance indicator in structural design, while the Decision Tree Classifier helps in optimizing the selection of I-section profiles based on the input parameters. This step showcases the integration of traditional structural engineering tools with emerging data-driven approaches.

The final phase involves evaluating the performance of the trained models using statistical metrics such as Mean Squared Error (MSE) and the R-Squared Score (R^2). These metrics help quantify the models' predictive accuracy and reliability. The results provide valuable insights into the feasibility and effectiveness of incorporating machine learning in structural analysis, highlighting its potential to enhance decision-making in engineering design processes.

Figure 3.1 illustrates a three-phase methodology for integrating machine learning into structural analysis using ETABS data. Phase 1 defines input parameters, runs ETABS simulations, and validates outputs. Phase 2 prepares the dataset, applies a Decision Tree Classifier for I-section prediction and regression models for displacement estimation. Phase 3 evaluates model performance using metrics like MSE and R^2, with visualizations to assess accuracy and relevance in structural design.

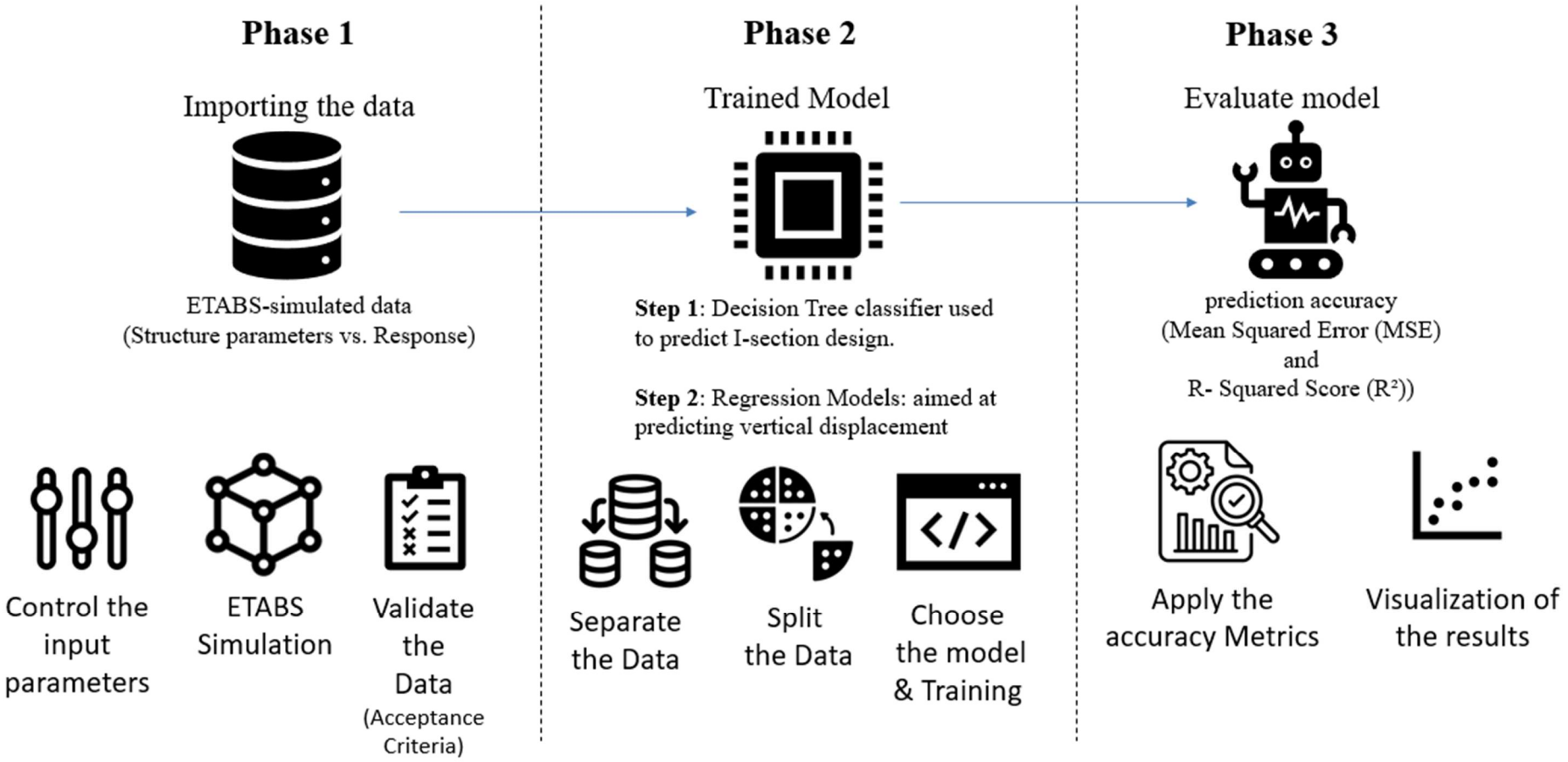


Figure 3.1: Data-Driven 3-Phase methodology for I-Section Classification and Displacement Prediction.

3.1 Data Generation and Preprocessing

ETABS software is used to generate the dataset, while a custom Python bot automates the retrieval of I-section designs and analysis results (U3) for various configurations of modulus of elasticity (E), span length (L), and applied load (P) without repetition. The extracted data is stored in an Excel file for further analysis. The Python script systematically generates different structural configurations by combining discrete values of E, P, and L, ensuring no duplicate cases. Table 3.1 presents the discrete values used for this process.

Table 3.1: Input Data for Structural Simulation.

Parameters	**values**	**Unit**
Steel properties	Modulus of Elasticity E: 200	GPa
	Yeilding Stress Fy: 344	MPa
Lengths	1, 1.1, 1.2, …. 5.8, 5.9, 6	m
Point Loads	1, 1.1, 1.2, …. 5.8, 5.9, 6	KN

The process illustrated in Figure 3.2 demonstrates an automated workflow for generating a structural analysis dataset using ETABS and a Python-based scripting tool. The workflow begins by defining key structural parameters such as modulus of elasticity (E), beam length (L), and applied load (P). These parameters are input into a Python script, referred to as the "Python Robot," which automates the communication with ETABS. The script sends the parameters to ETABS, where it initiates the modeling of a cantilever beam, performs the structural analysis, and starts the design process. ETABS then returns the preliminary design results, including the selected W-shape steel section. If necessary, ETABS re-runs the analysis based on the selected design to validate its adequacy. Both the design and analysis results are retrieved and linked to their corresponding input parameters. These paired data entries -structural features and their associated ETABS outputs- are then compiled into a comprehensive simulated dataset. This dataset can serve as a valuable resource for data-driven structural design, enabling further analysis, machine learning model training, or optimization studies. Look to appendix A to see the full python script.

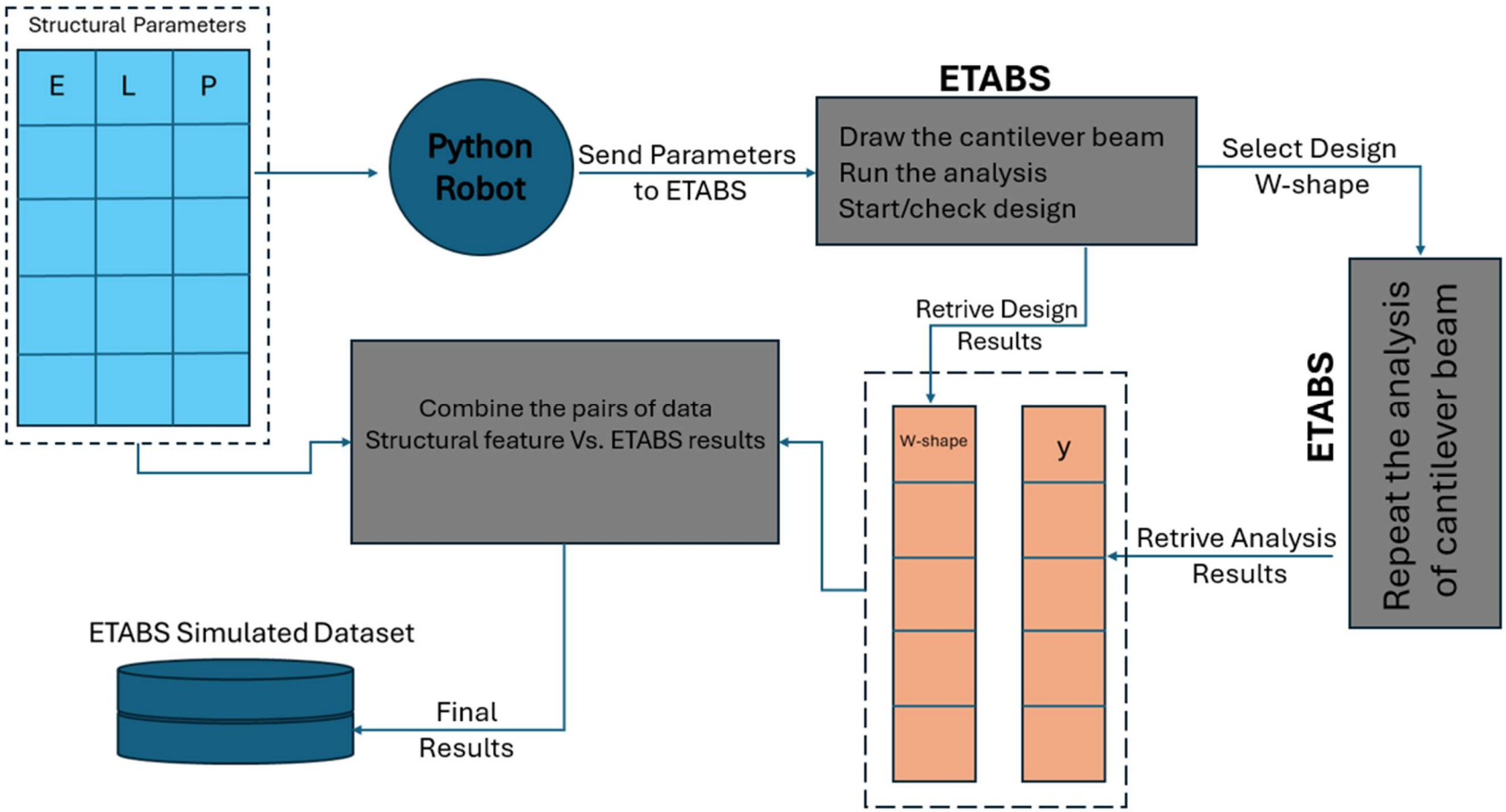


Figure 3.2: Automated Workflow for Cantilever Beam Analysis and Design Using Python and ETABS.

This flowchart in Figure 3.3 outlines the process for automating the analysis and design of cantilever beams using the ETABS API. The process begins with preparing the ETABS model by defining material properties and an auto-select list of W-shaped steel sections. A nested loop structure iterates over different values of beam lengths (L) and point loads (P), with counters i and j controlling the iterations.

For each combination of L_i and P_j the model enters a processing phase using the ETABS API. It starts by creating a new ETABS model and deleting any existing frames. A new cantilever beam is then drawn by defining its geometry and applying boundary conditions: fixed at the start and free at the end. A point load is applied at the free end, and the model is analyzed and designed.

The output includes the selected W-section (W-shape) and the vertical displacement (U3) at the tip of the cantilever. These results, along with the corresponding E, L, and P values, are saved in Excel as labeled data for training machine learning models. The loop continues until all parameter combinations are processed, after which the automation ends. This procedure ensures the efficient creation of a labeled dataset linking input parameters to structural responses.

Key steps of process for automating the analysis and design of cantilever beams using the ETABS could be summerize into main steps include:

- Establishing connection with ETABS and initializing a new model.
- Deleting existing frame elements.
- Drawing a cantilever beam with specific restraints and applied point loads.
- Running structural analysis and extracting design results (selected W-shape and displacement).
- Exporting results to Excel for labeling and documentation.

Look to appendix A to see the full python script.

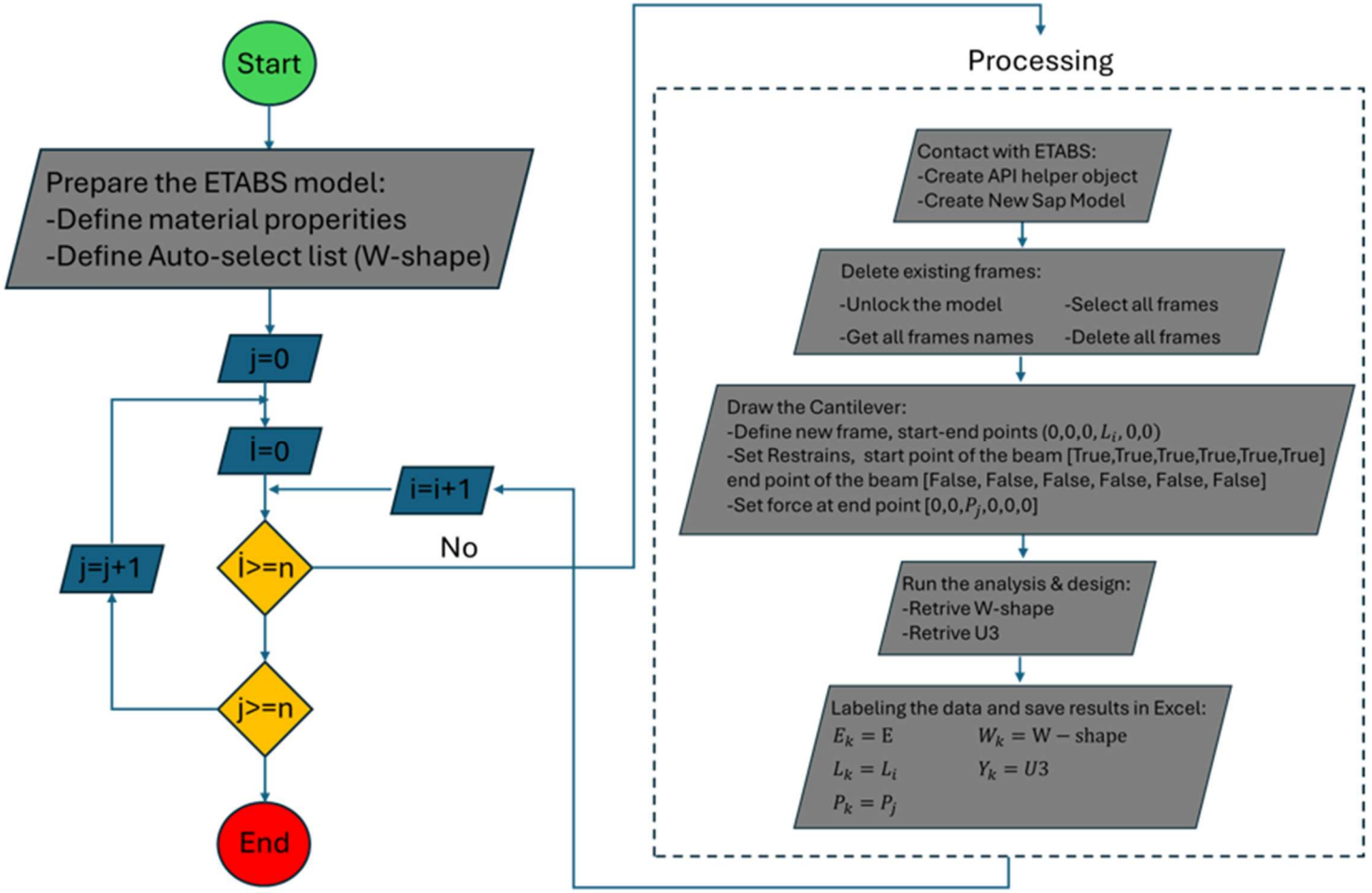


Figure 3.3: Automated Cantilever Beam Analysis and Design Using ETABS API

A new model was created in ETABS, where material properties (assumed constant E and Fy) and I-section frame auto-selection were defined, as shown in Figures 3.4 and 3.5. To automate data retrieval, a Python robot was developed using an API helper object, enabling a connection to the SapModel and extracting the ETABS model name. Finally, lists of lengths and loads were defined.

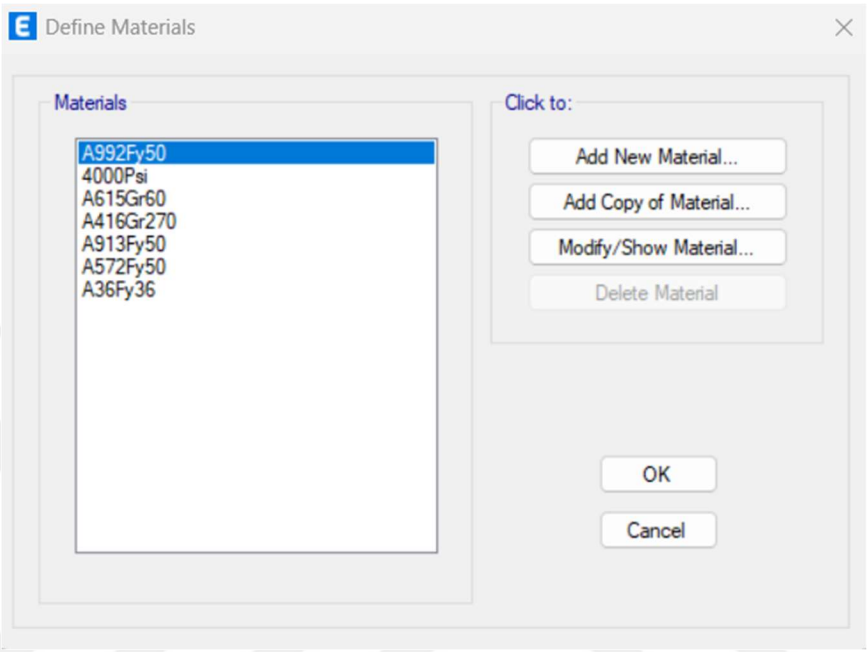


Figure 3.4: Define material properties.

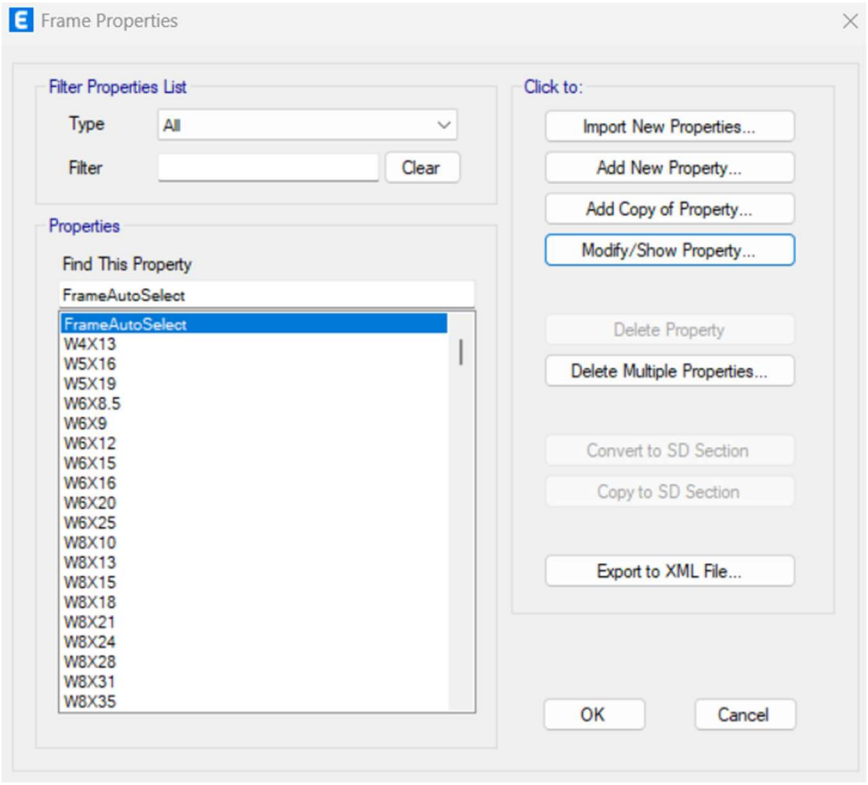


Figure 3.5: Define frame properties.

The next step involved passing the length and loads into the create_model function. This function was used to unlock the SapModel and delete all existing structures, set the present units to KN-m, and add a cantilever beam with the given length using SapModel.FrameObj.AddByCoord. The start point (0, 0, 0) and end point (length, 0, 0) were defined, and the frame assigned to Auto-select as shown in Figure 3.6 below.

Figure 3.6: Extrude view cantilever beam.

This function also applied restraints to the beam using SapModel.PointObj.SetRestraint, with (True, True, True, True, True, True) assigned to the fixed end and (False, False, False, False, False, False) to the free end. Additionally, SapModel.PointObj.SetLoadForce was used to assign the load as (0, 0, -load, 0, 0, 0) at the free end of the cantilever, as shown in Figure 3.7 below.

Figure 3.7: Assign point load.

After that, the function was built to run the analysis and design for the given structure features and retrieve I-section designs using SapModel.DesignSteel.GetSummaryResults_3. While SapModel.Results.JointDispl was used to return the displacement and rotation values (U1, U2, U3, R1, R2, R3) as shown in figure 3.8. The I-section design and vertical displacement of the cantilever (U3) were saved in an Excel sheet, corresponding to the E, L, and P parameters.

Figure 3.8: Retrieving displacement value from ETABS.

Another function was built to generate the dataset in sessions. The function iterated through all parameter values and repeated all the above steps. Each structure configuration was assigned an index and stored in a single row in the Excel sheet as a pair of data, with inputs corresponding to outputs. The variables tracked include input or independent features included the structure parameters: Modulus of Elasticity, Length, and Load, while the outputs or dependent variables were I-section and vertical displacement (U3) retrieved from ETABS. The loop terminates once all combinations have been processed, facilitating efficient parametric studies for cantilever beam behavior. The figure 3.9 represents how the dataset was stored in Excel. Each row corresponds to a unique structural configuration, indexed from 1 to 2601, and stored as a pair of input (E, L, P) and output (W-shape, y) values.

Lable	E	L	P	W-shape	y
1					
2					
3					
⋮	⋮	⋮	⋮	⋮	⋮
2600					
2601					

Figure 3.9: Structured Dataset for Parametric Study of Cantilever Beam Behavior.

3.2 Data Validation

The lists presented in Table 3.1 contain various structural parameters -namely E, L, and P- which serve as the basis for generating a wide range of structural configurations. In this setup, the modulus of elasticity (E) was held constant, while both the span length (L) and applied load (P) were varied across 51 distinct values each, resulting in a total of 2,601 unique parameter combinations. These combinations were saved in an Excel file, with each row representing a single configuration. Each structural parameter corresponds to key structural features, and a unique index (label) was assigned to every entry for use in supervising machine learning models. The design criterion was based on the allowable displacement (Δ_allowable), and each configuration was evaluated accordingly, as shown in Table 3.2 For structures that failed the criterion, the displacement value was set to zero.

Table 3.2: Acceptance Criteria based on Allowable displacement.

If	Results	Displacement	No. of data rows
$\boldsymbol{\delta_{actual} \leq \delta_{allowable}}$	Pass	$\delta = \delta_{ETABS}$	2601
$\boldsymbol{\delta_{actual} > \delta_{allowable}}$	Fail	$\delta = 0$	0
	Total		2601

Table 3.2 outlines the acceptance criteria for evaluating structural performance based on allowable vertical displacement. The evaluation is based on a comparison between the actual displacement ($\boldsymbol{\delta_{actual}}$) obtained from ETABS and a predefined allowable displacement limit ($\boldsymbol{\delta_{allowable}}$). If the actual displacement is less than or equal to the allowable limit, the structural configuration is considered acceptable, and the corresponding displacement value from ETABS is retained. Conversely, if the actual displacement exceeds the allowable threshold, the configuration is marked as a failure, and the displacement value is set to zero. In this study, all 2601 structural configurations met the allowable displacement requirement, resulting in a 100% pass rate and no failed cases. This confirms that all modeled beam configurations performed within acceptable limits for vertical displacement.

3.3 Dataset Preparation

Data preprocessing involved separating the dataset into dependent (what we want to predict) and independent variables (features that help predict the outcome) to prepare it for machine learning analysis.

Dependent Variable:

- U3 is the vertical displacement
- Selected design of I-section

Independent Variables:

- P is the load at the free end.
- *L* is the length of the cantilever.
- E is the modulus of elasticity.

The complete dataset was then split into two subsets: a training set and a testing set. An 80:20 ratio was used -a common practice in machine learning- where 80% of the data was used to train the model, and the remaining 20% was used to evaluate its predictive performance. This division ensures that the model can be tested on unseen data to assess its generalization ability. The distribution of this split is illustrated in Figure 5.7 below.

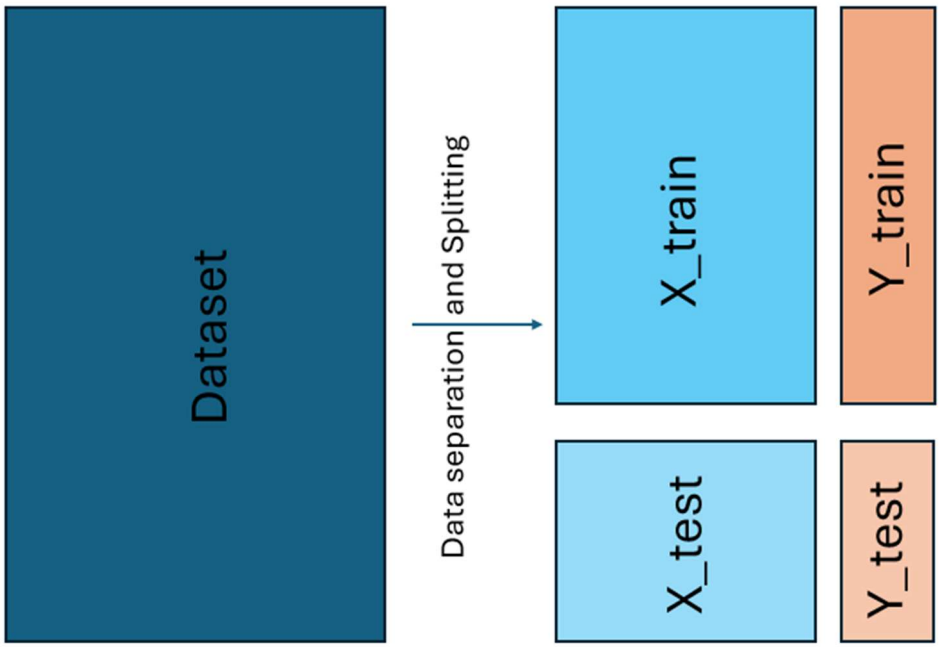


Figure 3.7: Data separation and splitting.

3.4 Machine Learning Model Development

This study primarily focused on two categories of supervised machine learning techniques. The first involved classification, where a Decision Tree Classifier was employed to predict the optimal I-section design for cantilever beams. The second approach focused on regression, aiming to estimate the vertical displacement of cantilever beams. For this purpose, three regression models were implemented: Linear Regression, Decision Tree Regressor, and Random Forest Regressor. Each model was evaluated for its effectiveness in capturing the relationship between the input parameters and the resulting displacement.

3.4.1 Classification Model

Classification algorithms are designed to predict discrete outcomes by assigning input data to predefined categories or classes. In this study, a Decision Tree Classifier was utilized to determine the most suitable I-section design for cantilever beams as shown in Figure 3.7. The prediction was based on key structural features, including the modulus of elasticity (E), cantilever length (L), and applied load (P). This approach enables the model to learn patterns in the data and accurately classify the optimal section design under varying structural conditions. Look to appendix B to see the full python script of Decision Tree Calssifier model.

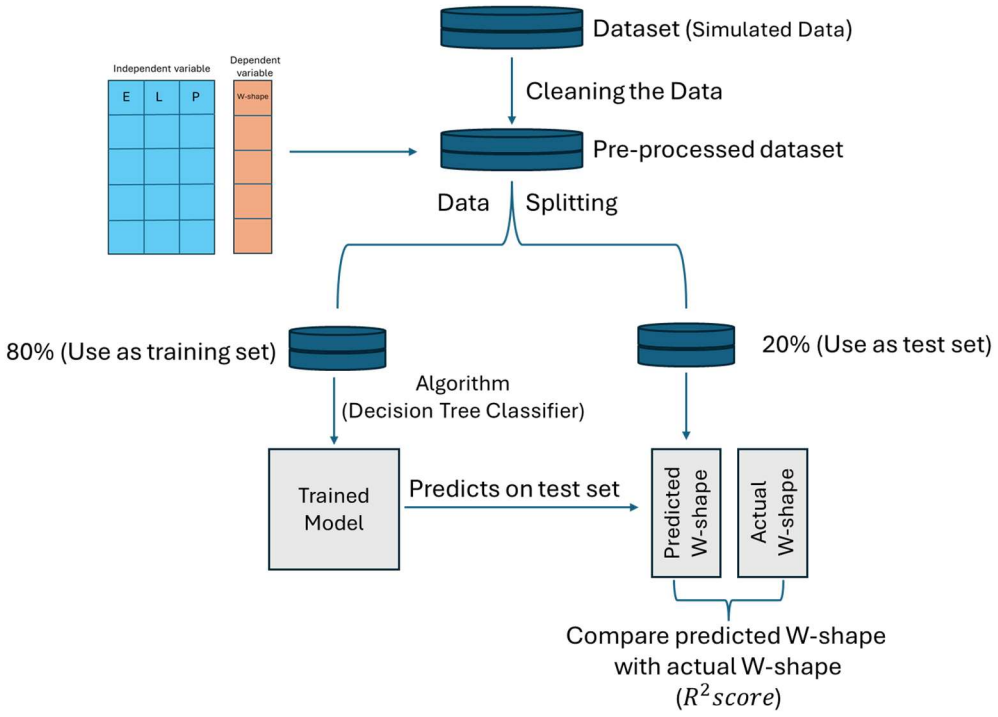


Figure 3.7: Workflow of I-Section Classification Using Decision Tree Classifier.

3.4.2 Regression Model

Regression algorithms are used to predict continuous numerical outcomes by modeling the relationship between dependent and independent variables. In the context of this study, regression techniques were applied to estimate the vertical displacement of cantilever beams as shown in Figure 3.8. The predictions were based on key structural features, including the modulus of elasticity, selected I-section design, span length, and applied load. By capturing the underlying patterns within these parameters, the regression models enable accurate and efficient prediction of displacement behavior under varying loading conditions. Look to appendix C & D to see the full python script of Regression models.

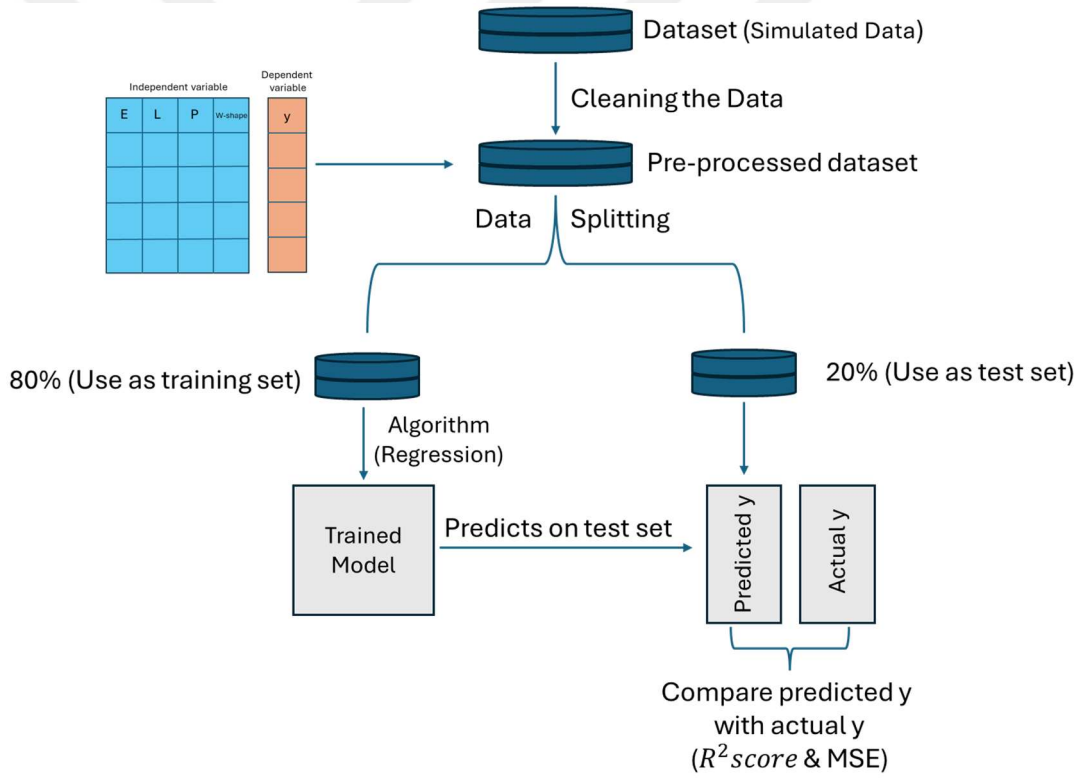


Figure 3.8: Workflow for Regression-Based Prediction of Cantilever Beam Displacement Using ETABS Simulated Data.

3.4.3 Model Training

Machine learning (ML) models are trained using paired datasets, where X represents the input features and Y represents the corresponding target values. During training, the model learns patterns and relationships from the X_train and Y_train data. Once the training process is complete, the model's performance is evaluated on a separate

testing set, X_test and Y_test, which the model has not seen during training. This ensures an unbiased assessment of the model's ability to generalize to new, unseen data [36]. Look at Figure 3.9.

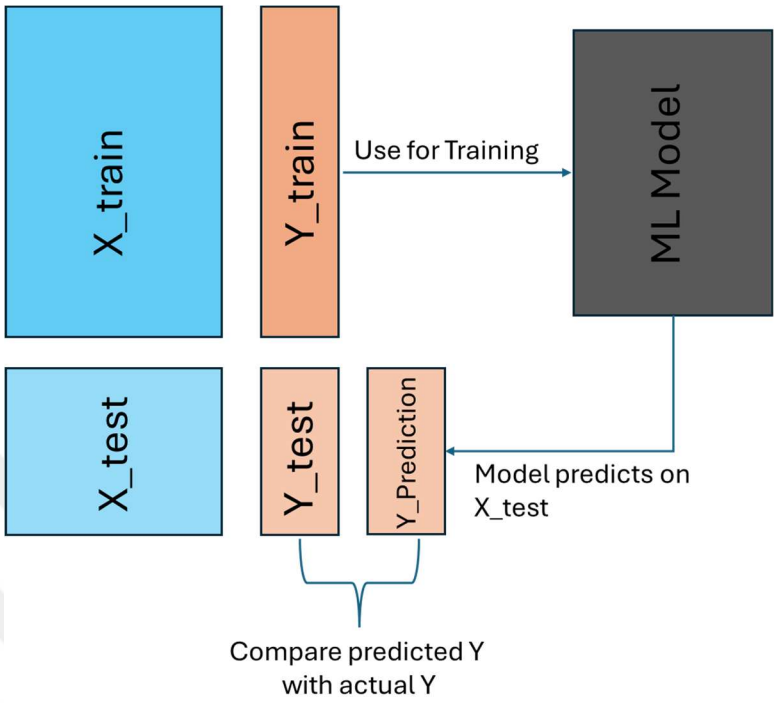


Figure 3.9: ML models training and performance evaluation.

3.4 Model Evaluation

Assessing the model's performance using appropriate metrics, such as mean squared error (MSE) or R-squared, to determine how well it predicts displacement.

3.4.1. Mean Squared Error (MSE)

Calculates the average squared difference between the predicted values and the actual values. Lower MSE indicates better model performance.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \overline{y}_i\right)^2$$

- A lower MSE indicates better model performance, as it means that the model's predictions are closer to the actual values.
- The "goodness" or "badness" of an MSE value can vary widely depending on the scale and nature of the data. For example, an MSE of 34373 might be

acceptable for predicting house prices in millions of dollars, but it would be unacceptable for predicting temperatures in degrees Celsius.

- To interpret the MSE value more accurately, consider comparing it to other models or to a baseline model.

3.4.2 R-squared (R^2) Score

Measures the proportion of the variance in the dependent variable that is predictable from the independent variables. R^2 score closer to 1 indicates a better fit of the model to the data. R-squared is calculated using the following formula:

$$R2 = 1 - \frac{SSres}{SStot}$$

Where:

- SS_{res} is the residual sum of squares, the sum of squared differences between the actual and predicted values.
- SS_{tot} is the total sum of squares, the sum of squared differences between the actual values and the mean of the actual values.

Ready-made libraries such as scikit-learn in Python can be used to calculate R-squared directly from your model predictions and actual target values. R^2 ranges from 0 to 1, where:

- $R^2 = 1$: Perfect fit, meaning that all variations in the dependent variable are explained by the independent variables.
- R^2 close to 1: Indicates a high percentage of the variance in the dependent variable is explained by the independent variables, suggesting a strong relationship between them.
- R^2 close to 0: Indicates that the independent variables do not explain much of the variance in the dependent variable, suggesting a weak relationship.
- $R^2 < 0$: This is theoretically possible if the model performs worse than simply using the mean of the dependent variable to predict outcomes, but it is extremely rare in practice.

4 IMPLEMENTATION AND RESULTS

This chapter outlines the implementation process and presents the results obtained from the proposed methodologies. The approach is structured into a three-phase, data-driven workflow designed to support the design of I-sections and the prediction of vertical displacement of steel cantilever beam based on structure features such as, modulus of elasticity (E), span length (L), and applied load (P) using machine learning techniques. Key insights and observations from the results are also discussed.

Phase 1 focuses on data preparation, which includes three essential micro-steps. First, the input parameters are carefully controlled to define structural properties and loading conditions. Second, ETABS simulations are conducted to generate response data based on the defined inputs. Third, the simulated data is validated against predefined acceptance criteria to ensure the dataset's quality.

Phase 2 involves model development and training, consisting of three micro-steps. The process begins by separating the features and target variables. Next, the data is split into training and testing subsets. Finally, appropriate machine learning models are selected and trained. The case study is consists of two main steps: first, the Decision Tree Classifier is applied to predict the I-section selection; second, regression analysis is used to predict vertical displacement (U3).

Phase 3 is dedicated to model evaluation. Accuracy metrics were applied, specifically the Mean Squared Error (MSE) and the R-squared (R^2) score, to evaluate prediction accuracy. Additionally, the results are visualized through appropriate plots and graphs to better understand the model's behavior and predictive capabilities.

Figure 4.1 visually summarizes the three-phase methodology: data preparation, model development, and model evaluation. This workflow integrates ETABS simulation data with machine learning to enable efficient I-section design and accurate displacement prediction. Each phase builds on the previous one, ensuring a systematic and reliable process from data generation to predictive modeling and performance assessment.

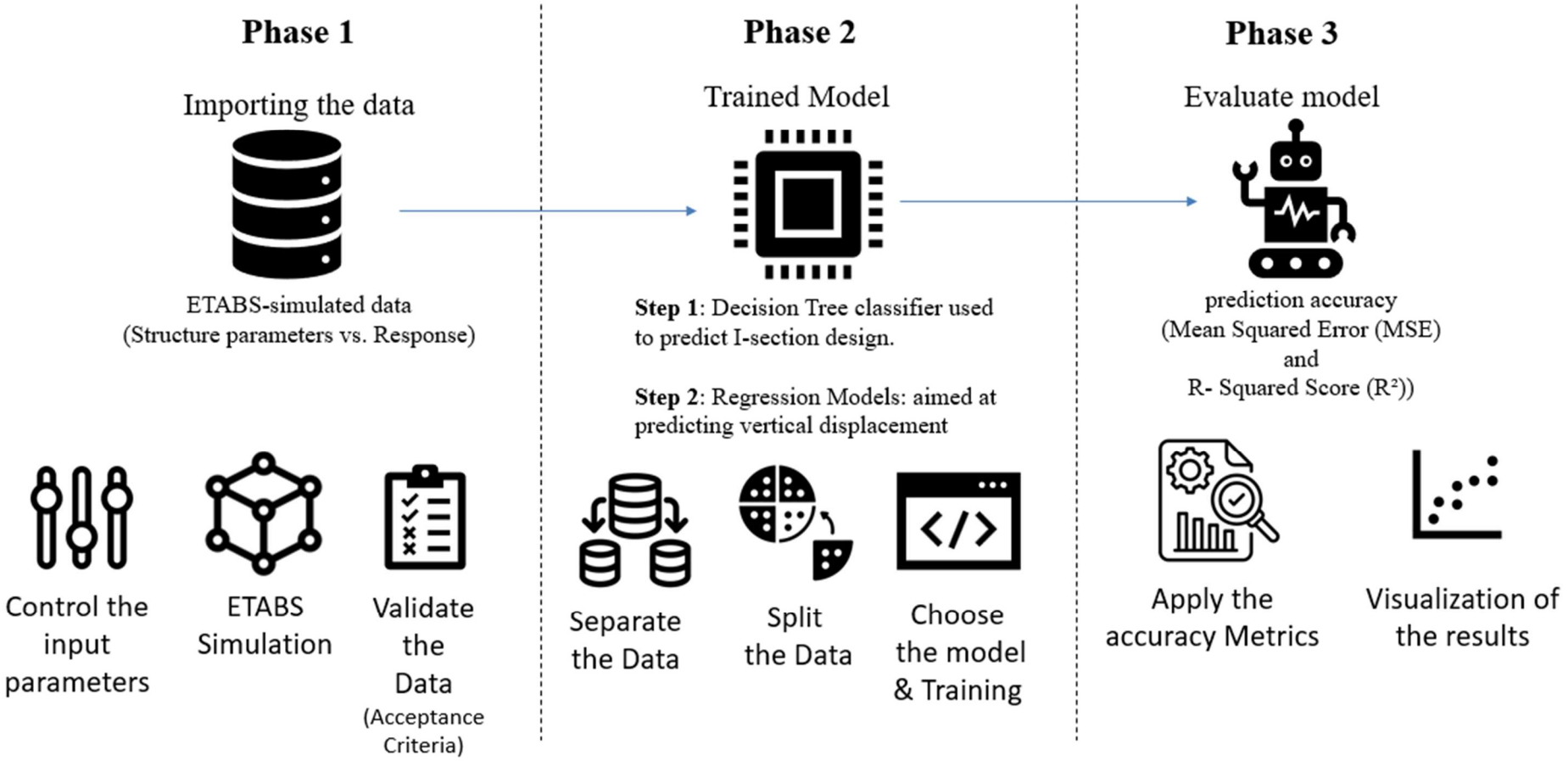


Figure 4.1: Three-Phase Data-Driven Approach for I-Section Selection and Displacement Prediction.

4.1 Phase 1: Data Generation

This phase focuses on data generation to discuss different structure configrations. This involves extracting analytical results through the ETABS API for multiple configurations, including different beam lengths, applied loads, and material properties. The goal of this phase is to generate a comprehensive and diverse dataset that accurately captures the structural response of steel cantilever beams.

First, the structure parameters were carfully controlled to define the structure properties and loading conditions. These features are modulus of elasticity (E), span length (L), and applied load (P). Table 4.1 presents the discrete values used for this features.

Table 4.1: Structure parameters that define the range of structure configrations.

Parameters	**values**	**Unit**
Steel properties	Modulus of Elasticity E: 200	GPa
	Yeilding Stress Fy: 344	MPa
Lengths	1, 1.1, 1.2, …. 5.8, 5.9, 6	m
Point Loads	1, 1.1, 1.2, …. 5.8, 5.9, 6	KN

Seconed, ETABS software is used to generate the dataset, while a custom Python bot automates the retrieval of I-section designs and analysis results (U3) for various configurations of modulus of elasticity (E), span length (L), and applied load (P) without repetition. The extracted data is stored in an Excel file for further analysis. The Python script systematically generates different structural configurations by combining discrete values of E, P, and L, ensuring no duplicate cases.

Figure 4.2 illustrates the automated workflow for cantilever beam analysis and design using Python and ETABS. This process was described in detail in the previous chapter, and the implementation code is provided in Appendix A. The workflow begins with input structure parameters which are processed through a Python script that interacts with ETABS. The corresponding structure responses, including the selected W-shape section and vertical displacement (y), are extracted as outputs. The final output of this automated process is a dataset (structure parameters vs. response).

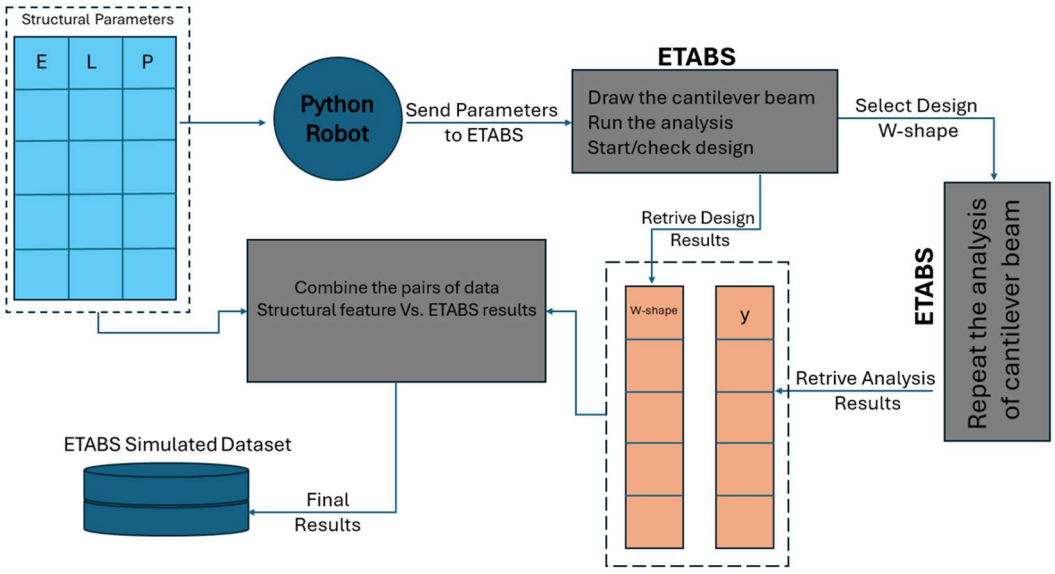


Figure 4.2: Data Generation Pipeline for Cantilever Beam Design via ETABS and Python Scripting.

In this setup, the modulus of elasticity (E) was held constant, while both the span length (L) and applied load (P) were varied across 51 distinct values each, resulting in a total of 2,601 unique parameter combinations. These combinations were saved in an Excel file, with each row representing a single configuration. Each structural parameter corresponds to key structural features, and a unique index (label) was assigned to every entry for use in supervising machine learning models.

Third, the simulated data is validated against predefined acceptance criteria to ensure the dataset's quality and qualified the data. The acceptance criterion was based on the allowable displacement (Δ_allowable), and each configuration was evaluated accordingly, as shown in Table 4.2 For structures that failed the criterion, the displacement value was set to zero.

Table 4.2: Qualification the data based on Allowable displacement.

If	Results	Displacement	No. of data rows
$\boldsymbol{\delta_{actual} \leq \delta_{allowable}}$	Pass	$\delta = \delta_{ETABS}$	2601
$\boldsymbol{\delta_{actual} > \delta_{allowable}}$	Fail	$\delta = 0$	0
	Total		2601

4.2 Phase 2: Model Development and Training

Phase 2 focuses on the development and training of machine learning models through three key micro-steps, systematically designed to prepare the data and optimize the performance of predictive algorithms. The first step involves separating the dataset into input features and target variables. The features include the structural parameters (elastic modulus, beam length, applied load), while the targets represent the desired outputs, such as the beam's displacement and cross-section designation.

The second step involves partitioning the data into training and testing subsets. This split is essential to ensure that the model can generalize well to unseen data, allowing for accurate performance evaluation during the testing phase. Typically, a standard ratio 80:20 is used to maintain a balance between training data sufficiency and evaluation integrity.

The third and final step in this phase is the selection and training of suitable machine learning models. In this case study, two primary tasks are addressed using distinct algorithms. First, the Decision Tree Classifier is employed to predict the most suitable I-shaped steel section (W-shape) based on the input parameters. This classification task helps automate the structural design process by mapping parameter combinations to discrete section labels. Second, a regression model is applied to predict the vertical displacement (U3) of the cantilever beam. This task involves estimating a continuous response variable and is crucial for assessing structural performance under loading conditions.

4.3 Phase 3: Model Evaluation

Phase 3 is dedicated to evaluating the performance and reliability of the trained models through applying accuracy metrics. To quantify the prediction accuracy two widely accepted evaluation metrics are utilized: the Mean Squared Error (MSE) and the R-squared (R^2) score. MSE measures the average of the squares of the prediction errors, providing insight into the model’s precision. A lower MSE indicates better performance. The R^2 score, on the other hand, indicates the proportion of the variance in the dependent variable that is predictable from the independent variables, with values closer to 1 denoting a stronger predictive model.

In addition to numerical evaluation, visual tools are employed to gain deeper insights into model behavior. Performance plots, such as actual vs. predicted graphs and residual plots, are generated to assess the quality of predictions and to identify potential overfitting or underfitting. These visualizations not only validate the model's effectiveness but also help in interpreting the influence of various features on the predicted outcomes.

Together, Phases 2 and 3 form the core of the machine learning integration in this study, transforming raw structural data into intelligent predictions that support automated design and analysis of cantilever beams.

4.4 Study Case

To validate the aims of this thesis, one case study was conducted. It consists of two main steps: first, the Decision Tree Classifier is applied to predict the I-section selection of a cantilever beam based on input parameters like modulus of elasticity, length and load; second, regression analysis is used to predict vertical displacement (U3) of the steel cantilever beam based on input parameters like beam length, load, modulus of elasticity, and W-shape. This approach assessed the machine learning model's effectiveness in recommending optimized design parameters based on performance criteria, demonstrating their potential in enhancing structural efficiency and accuracy.

4.3.1 Step 1

The Decision Tree Classifier is applied to predict the I-section selection of a cantilever beam based on input parameters like modulus of elasticity, length and load. Appendix B contains the complete Python code for the Decision Tree Classifier, along with the model predictions, their corresponding cantilever configurations, and the calculated R-squared (R^2) score.

The dataset consists of 2,601 rows of data. At this stage, the data was structured into input-output pairs, where E, L, and P served as independent features, and the I-section was the target variable as shown in Figure 4.3 below. A Decision Tree Classifier was employed to predict the I-section selection based on these parameters. To train and evaluate the model, the dataset was split into 2,080 training samples and 521 testing samples, ensuring a balanced distribution for effective learning and validation.

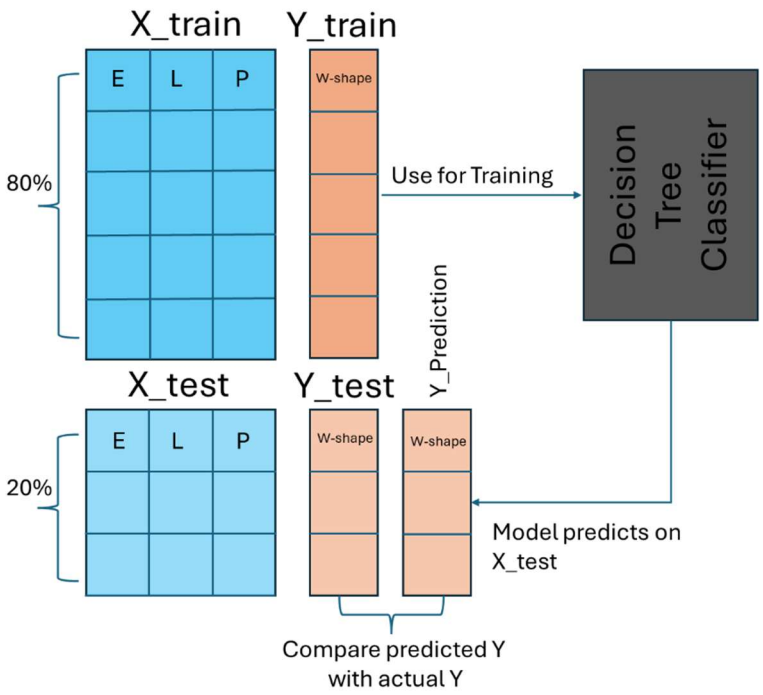


Figure 4.3: Dataset separation and splitting-first step.

Figure 4.4 illustrates the feature importance within the decision tree, highlighting the relative influence of E, L, and P on the model's predictions. Shows that steel grade (modulus of elasticity E) has no impact on the I-section design selection, as it kept constant at 200 GPa Throughout the study.

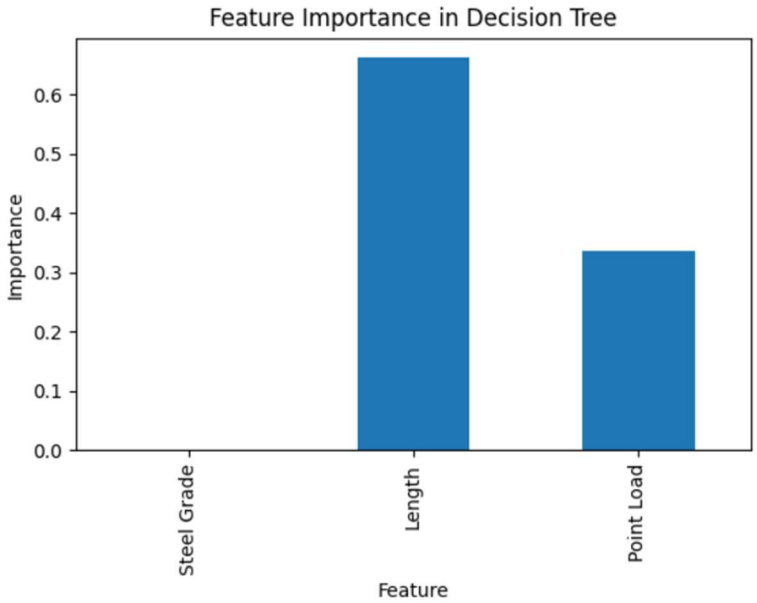


Figure 4.4: Feature importance within the decision tree

The performance of the Decision Tree Classifier was evaluated using the R^2 score, which exceeded 0.94, indicating a high accuracy in predicting the I-section design of the cantilever beam.

The scatter plot in Figure 4.5 illustrates the comparison between actual and predicted W-shape steel beam designs. Each point represents a design sample, with the x-axis denoting the actual W-shape design and the y-axis indicating the corresponding predicted design. The red dashed line represents the line of best fit, highlighting the correlation between actual and predicted values. Points below the line indicate under-designed predictions, which may require improvement to meet structural requirements. Conversely, points above the line represent over-designed predictions, which could lead to slightly higher costs.

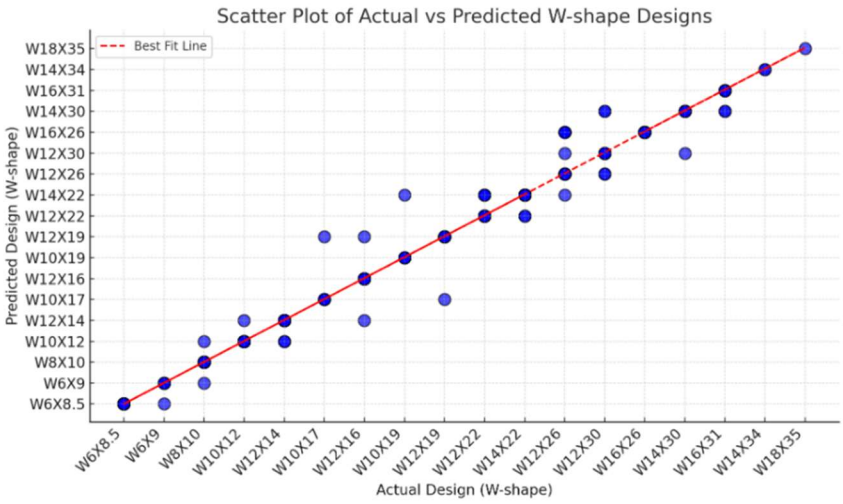


Figure 4.5: Comparison of Actual vs Predicted W-Shape Steel Beam Designs.

Table 4.3 presents a comparison between the results of ETABS and the clustering ML model predictions for a randomly selected structural configuration. This comparison aims to evaluate the performance of clustering ML models, specifically the Decision Tree Classifier, and their potential for integration into the design optimization process.

Table 4.3: I-section Design ETABS Vs. Decision Tree Classifier prediction.

Index	Length (m)	Load (KN)	ETABS results	ML Prediction
2152	4.5	2.2	W14X22	W14X22
117	2.7	2.4	W10X12	W10X12
168	3.3	4.3	W12X14	W12X14
801	2.2	1.9	W8X10	W8X10
1637	2.2	5.4	W10X12	W10X12

4.3.2 Step 2

Regression analysis is used to predict vertical displacement (U3) of the steel cantilever beam based on input parameters like beam length, load, modulus of elasticity, and W-shape. Appendix C includes the complete Python code for the Decision Tree Regressor, along with model predictions, corresponding cantilever configurations, true values, the R-squared (R^2) score, Mean Squared Error (MSE), and a true vs. predicted values chart.

A comparison of three types of regression models was conducted to evaluate their performance in predicting structural responses in this study. The models include Linear Regression, Decision Tree Regressor, and Random Forest Regressor. This comparative analysis aims to determine which algorithm performs best in predicting vertical displacement (U3) based on cantilever beam configurations. Appendix D contains the full Python code for all three models, their corresponding predictions, associated cantilever configurations, and the true vs. predicted values chart.

At this stage, E, L, W-shape, and P were used as independent features, with vertical displacement (y) as the target variable as shown in Figure 4.3 below. Three regression models -Linear Regression, Decision Tree Regressor, and Random Forest Regressor- were implemented to predict I-section selection based on these parameters. By comparing the predicted y with actual y using accuracy metrics, such as R-squared (R^2) score, and Mean Squared Error (MSE) the models were evaluated.

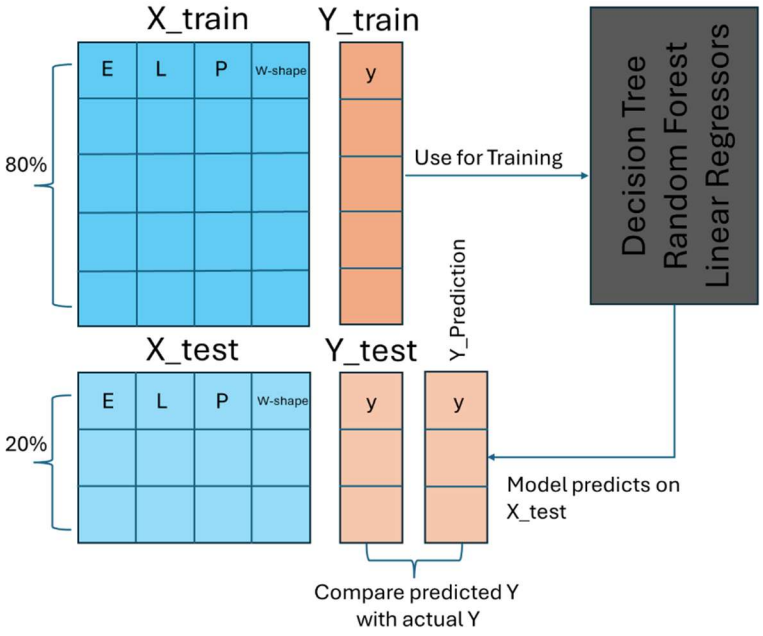


Figure 4.6: Dataset separation and splitting-second step.

Figure 4.7 presents a scatter plot comparing the predicted and actual displacement values across three machine learning models. The Decision Tree Regressor and Random Forest Regressor achieved a near-perfect fit, with best-fit line slopes close to 1. In contrast, Linear Regression showed a slightly lower accuracy, with a slope just above 0.9, indicating a fair fit.

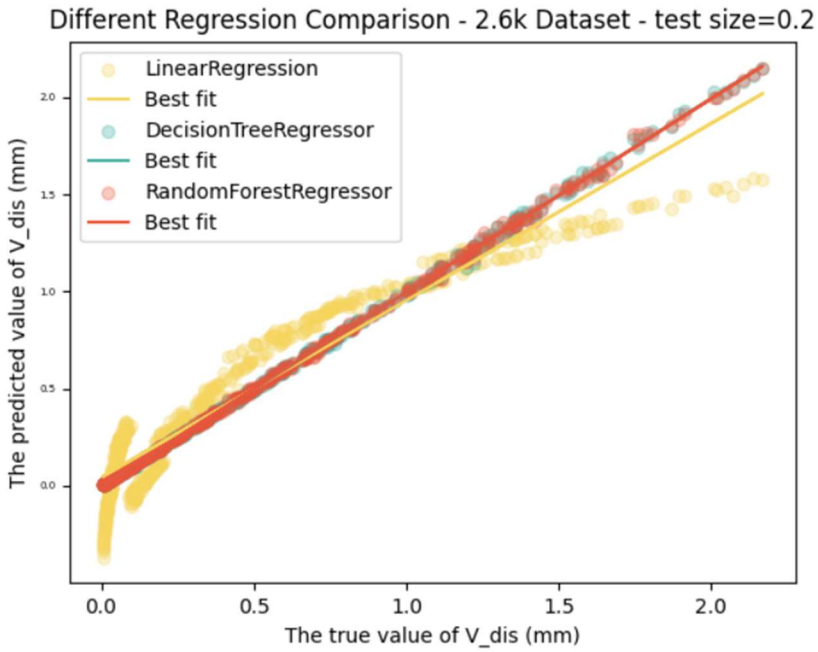


Figure 4.6: Results of 2.6k row dataset – test size=0.2

Figure 4.7 indicates that the R^2 score are just above 0.9, indicating a poor fit for linear regression, while the score is exactly 1, showing a perfect fit for the decision tree regressor and random forest regressor.

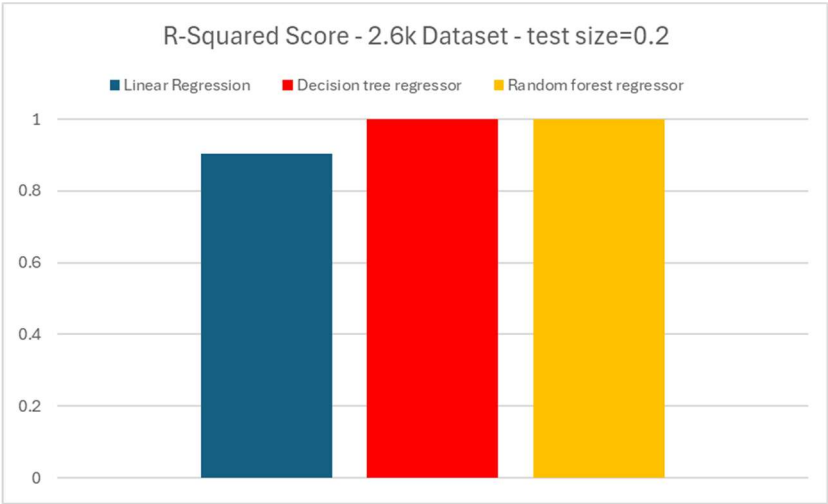


Figure 4.7: R – squared (R^2 Score) – 4k dataset.

Figure 2.8 indicates that the MSE for linear regression is significantly higher than the decision tree regressor and random forest regressor indicating a relatively poor fit for linear regression compared to other models.

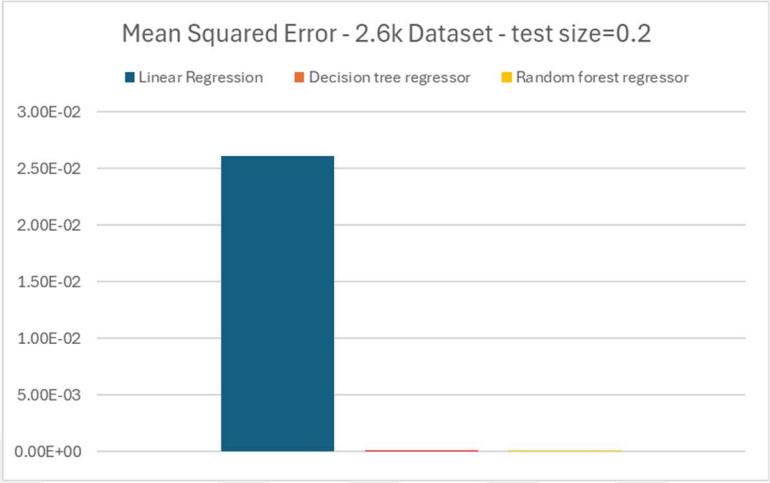


Figure 4.8: Mean Squared Error (MSE) – 4k dataset.

Table 4.3 presents the regression predictions compared to the true values. The test values represent the true values, while the first, second, and third predictions correspond to the outputs of the Linear Regression, Decision Tree Regressor, and Random Forest Regressor models, respectively.

Table 4.3: Different regression prediction comparison.

Index	Y_test	Y_pred1	Y_pred2	Y_pred3
2152	0.533	0.746625	0.543	0.53757
117	0.103	-0.092249	0.106	0.10768
168	0.272	0.358330	0.276	0.27212
801	0.049	0.121568	0.048	0.04948
1637	0.097	-0.045939	0.095	0.09239

5 DISCUSSIONS

This chapter presents the results and discussion of the study, focusing on the performance and implications of the machine learning models applied to the structural analysis of steel cantilever beams. It begins with a comparison between the predicted and actual results, highlighting the accuracy of each model. The chapter then evaluates the quality of the predicted I-section designs by categorizing them as underdesigned, probably well-designed, or overdesigned. This is followed by an analysis of insights gained from the results, including patterns and model behavior. The chapter concludes with a summary of key findings and reflects on the limitations of the study while offering recommendations for future research.

5.1 Results Comparation

Table 5.1 shows a comparison between I-section designs from ETABS simulations and those predicted by the Decision Tree Classifier model, based on randomly chosen structural setups. The comparison focuses on beam length and applied load corospending to the W-shap of each setup. In most cases, the machine learning model predictions closely matches the ETABS results, showing that it can reliably predict the right I-section. This suggests the model is not only accurate but also practical for helping engineers make quick and confident design choices without always relying on time-consuming simulations.

Table 5.1: Comparison of ETABS I-Section Design Results and Decision Tree Classifier Predictions.

Index	Length (m)	Load (KN)	ETABS results	ML Prediction
2152	4.5	2.2	W14X22	W14X22
117	2.7	2.4	W10X12	W10X12
168	3.3	4.3	W12X14	W12X14
801	2.2	1.9	W8X10	W8X10
1637	2.2	5.4	W10X12	W10X12

Table 5.2 presents a comparison between the vertical displacement results obtained from ETABS (Y_test) and the predictions from three different machine learning regression models: Linear Regression (Y_pred1), Decision Tree Regressor (Y_pred2), and Random Forest Regressor (Y_pred3). The table highlights how closely each model's predictions align with the actual ETABS results, illustrating the superior accuracy of tree-based models over linear regression for structural displacement prediction.

Table 5.2: Comparison of ETABS Results and Machine Learning Regression Model Predictions.

Index	Y_test	Y_pred1	Y_pred2	Y_pred3
2152	0.533	0.746625	0.543	0.53757
117	0.103	-0.092249	0.106	0.10768
168	0.272	0.358330	0.276	0.27212
801	0.049	0.121568	0.048	0.04948
1637	0.097	-0.045939	0.095	0.09239

5.2 Underdesigned and Overdesigned

The prediction results for I-section design fall into three categories: properly designed, underdesigned, and overdesigned. A structure is considered properly designed when the predicted design matches the actual design in terms of stiffness. If the predicted design is stiffer than the actual design, it is classified as overdesigned; if it is less stiff, it is underdesigned. Overdesigned structures tend to be safer but more expensive, whereas underdesigned structures may pose safety concerns and require improvement -possibly to be addressed in future studies.

he scatter plot in Figure 5.1 illustrates a comparison between the I-section designs (W-shapes) obtained from ETABS (Actual Design) and those predicted by the Decision Tree Classifier (Predicted Design). The x-axis represents the actual designs, while the y-axis shows the predicted designs. The W-shapes are arranged in a specific order, from the least stiff to the most stiff I-sections. The best-fit line indicates properly designed cantilever beams. Data points above the line represent overdesigned structures, while those below the line indicate underdesigned ones.

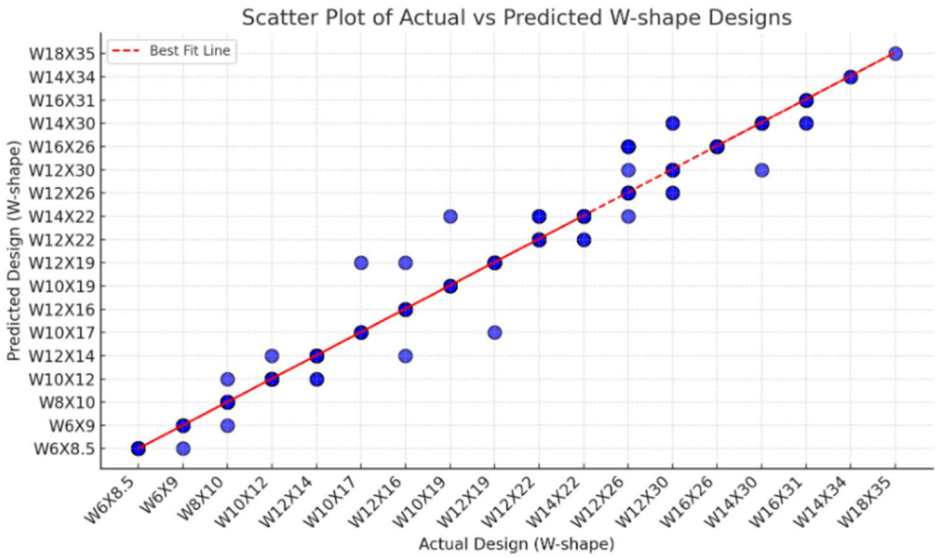


Figure 5.1: underdesigned and overdesigned structure plot.

The pie chart in Figure 5.2 shows the distribution of design categories. The testing set consists of 521 samples, which account for 20% of the total dataset (2,601 rows). Out of these, 490 structures were properly designed, 15 were underdesigned, and 16 were overdesigned. This breakdown highlights that the majority of the predictions closely matched the actual designs, with only a small portion falling outside the ideal range.

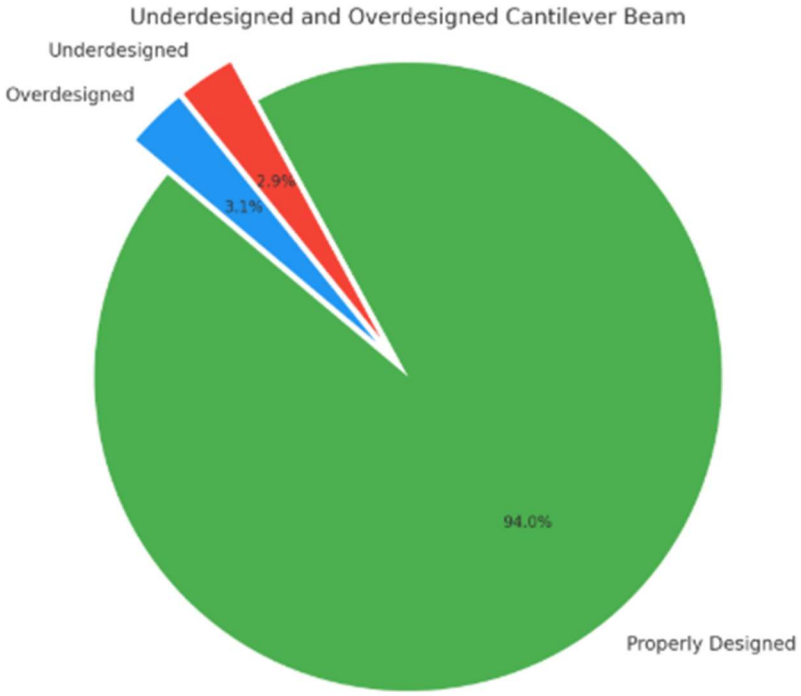


Figure 5.2: percentage of underdesigned and overdesigned structure pie chart.

5.3 Results Insights

The scatter plot in Figure 5.3 presents a design chart that maps predicted W-shape steel beam selections based on varying span lengths and point loads. Each dot on the chart represents a specific design scenario, with the x-axis indicating the span length in meters and the y-axis showing the applied point load in kilonewtons (kN). The color of each point corresponds to the predicted W-shape beam, as identified in the legend.

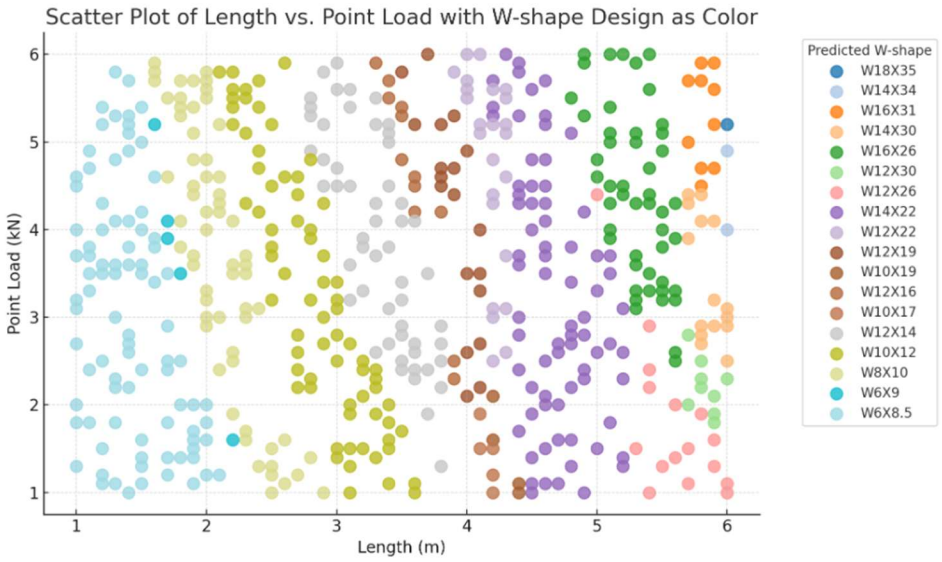


Figure 5.3: Scatter Plot of Span Length vs. Point Load Colored by Predicted W-Shape Steel Beam Design.

The chart reveals clear trends in how different W-shape sections are chosen according to structural demands. Heavier and larger beam sections are predominantly selected for scenarios involving longer spans and higher point loads, while smaller, lighter sections appear in regions with shorter spans and lower loads.

As a visual tool, this design chart aids in quickly understanding the model's decision-making and provides engineers with a reference for preliminary beam selection based on load-span requirements. It also offers an opportunity to identify potential outliers or inconsistencies in the prediction logic.

Figure 5.4 divides the chart into regions that provide approximate design guidance. The dashed lines represent estimated boundaries for selecting specific W-shape sections based on point load and span length. It is important to note that this chart is intended solely for illustrating results and is not to be used as an actual design chart.

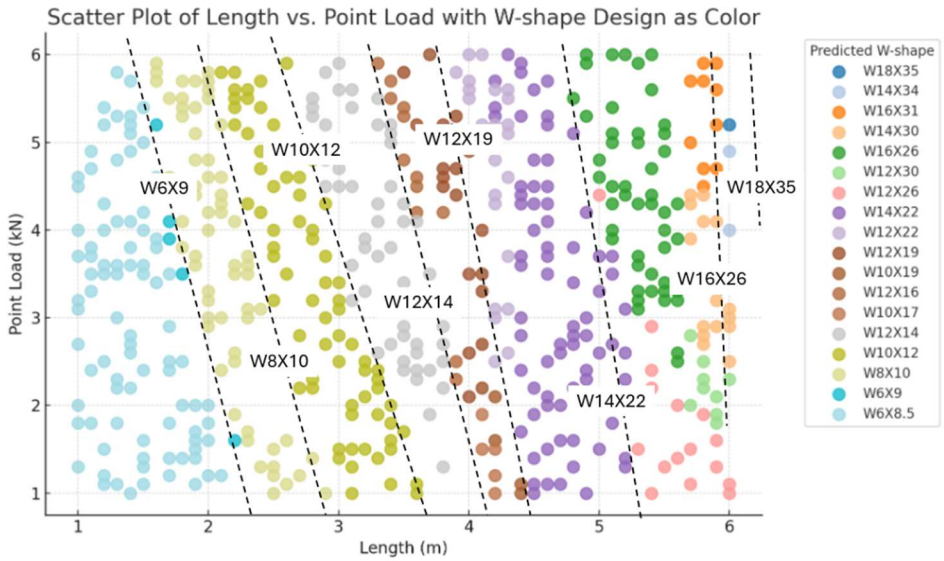


Figure 5.4: Scatter Plot of Beam Length vs. Point Load with Predicted W-Shape Designation.

The scatter plot in figure 5.5 illustrates the relationship between the length of a cantilever beam (in meters) and its corresponding vertical displacement at the free end (U3, measured in millimeters). Each data point represents a different loading and structural design scenario. Two datasets are compared: the predicted U3 values, shown as red circular markers, and the actual U3 values, shown as blue circular markers. This side-by-side visualization provides a clear comparison of model accuracy across a wide design space.

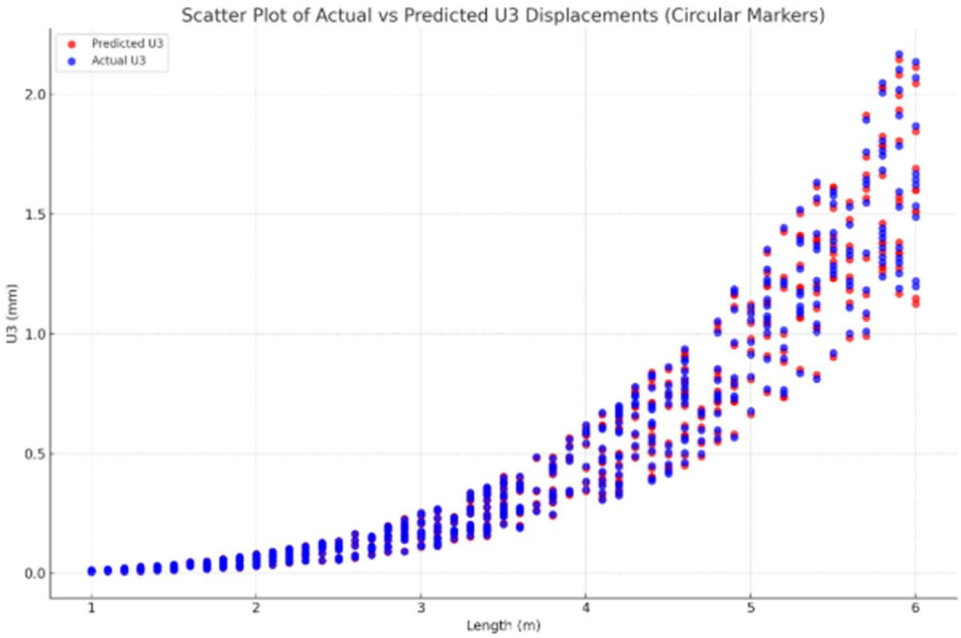


Figure 5.5: The relationship between the length of a cantilever beam (in meters) and its corresponding vertical displacement at the free end (U3, measured in millimeters).

Red circular markers indicate the predicted U3 values obtained from Decision Tree Regressor, while blue circular markers represent the actual U3 values obtained from ETABS simulation results. Each data point represents a different loading and structural design scenario. Therefore, the points are not concentrated along a single curve but rather form vertical clusters or regions at discrete beam lengths. These regions represent multiple design cases for the same length -where each case might involve different point loads and W-shape (I-section) profiles. For example, a 2-meter beam might have several design configurations, each with a unique combination of load and section type, resulting in a range of U3 displacement values. This explains the spread of points at each vertical band.

As the length of the cantilever increases, both predicted and actual U3 values generally show an upward trend, which aligns with structural mechanical principles. Longer beams are inherently more flexible and thus experience larger vertical displacements under the same loading conditions. This trend reinforces the expectation that displacement grows non-linearly with increased span, especially in cantilever systems where bending effects dominate.

By comparing the positions of the red and blue markers, one can evaluate how well the model approximates real-world behavior. In many instances, the predicted U3 values are close to the actual values, indicating a good level of accuracy. The proximity of the red and blue points suggests that the model captures the general trend of displacement with respect to length effectively. However, in some regions, noticeable differences between predicted and actual values appear, indicating potential areas where the model might be underfitting or overfitting.

5.4 Conclusion

The Decision Tree Classifier successfully predicted the appropriate W-shape based on structural features. The steel properties, cantilever length, and point load were input into the trained Decision Tree model, which outputs the corresponding W-shape. The model's performance was evaluated by comparing its predictions with results from ETABS, yielding an R^2 score slightly above 0.94, indicating a good fit.

Regression-based machine learning models were employed to predict vertical displacement based on structural features. The input parameters included steel

properties, cantilever length, W-shape, and point load. The trained regression models output the vertical displacement (U3). Model evaluation showed that linear regression had an R^2 score just above 0.9, indicating a fair fit, whereas the Decision Tree Regressor and Random Forest Regressor achieved an R^2 score of 1, demonstrating a perfect fit.

The dataset's dependent and independent variables were generated in a structured sequence. First, independent parameters were arbitrarily selected and input into ETABS to design the I-section and determine the appropriate W-shape. Once the W-shape was identified, the vertical displacement (U3) was retrieved from ETABS and compared against allowable deflection limits. This sequence was crucial to validate the data.

The machine learning models demonstrated strong predictive capabilities compared to ETABS-derived results. However, ETABS does not always yield the most economical W-shape due to various design preferences and parameters. This is not a concern because if data from different sources (such as STAAD, ROBOT, or Excel) were used, the model could be retrained on the new datasets and would perform similarly to the given study case.

5.5 Limitation and Recommendations

Hence, it is strongly recommended that structural engineers and researchers explore the integration of Machine Learning (ML) to enhance structural engineering processes. This integration will enable the automation of predictive analyses based on structural parameters, improving efficiency, safety, and cost-effectiveness in design. Furthermore, future efforts should focus on addressing the limitations of ML models, ensuring data reliability, and refining model accuracy to optimize their practical application. Successfully overcoming these challenges represents a crucial step toward fully leveraging data-driven methodologies in structural analysis and design.

The study acknowledges a key limitation. It was focused exclusively on steel cantilever beams subjected to a point load at the free end. While this approach was sufficient for predicting vertical displacement and I-section design, it may not fully represent other structural configurations. Therefore, further research should consider a broader range of structural systems, including beams, frames, and composite structures, subjected to various loading conditions. Expanding the scope in future studies will help validate the

findings and enhance the generalizability of machine learning applications in structural engineering.

Additionally, future research should extend beyond simple beam structures to address more complex systems, such as built-up girders and columns. Studying these components will better reflect real-world conditions and enhance the practical application of machine learning techniques. This approach will ultimately lead to more robust and versatile predictive models.

To ensure data reliability in structural engineering applications, it is crucial for experts and researchers to engage in specialized workshops and collaborative sessions. These gatherings provide opportunities to standardize design preferences, clarify key parameters, and align on best practices. By combining practical experience with academic insights, workshops can help build high-quality datasets that better reflect real-world conditions. This approach enhances the overall accuracy and reliability of machine learning models used in structural analysis and design.

REFERENCES

[1] Deka, P. C. *(2020). A Primer on Machine Learning Applications in Civil Engineering. CRC Press.*

[2] Plevris, V., Ahmad, A., & Lagaros, N. D. (Eds.). (2023). *Artificial Intelligence and Machine Learning Techniques for Civil Engineering*. IGI Global.

[3] Kopec, D. (2019). *Classic Computer Science Problems in Python*. Manning Publications.

[4] Chitkeshwar, A. (2024). Revolutionizing Structural Engineering: Applications of Machine Learning for Enhanced Performance and Safety. *Archives of Computational Methods in Engineering*.

[5] Teymori Gharah Tapeh, A., & Naser, M. Z. (2022). Artificial Intelligence, Machine Learning, and Deep Learning in Structural Engineering: A Scientometrics Review of Trends and Best Practices. *Archives of Computational Methods in Engineering.*

[6] Málaga-Chuquitaype, C. (2022). Machine Learning in Structural Design: An Opinionated Review. *Frontiers in Built Environment, 8*, 815717. https://doi.org/10.3389/fbuil.2022.815717

[7] Plevris, V., & Papazafeiropoulos, G. (2024). AI in structural health monitoring for infrastructure maintenance and safety. *Infrastructures, 9*(12), 225. https://doi.org/10.3390/infrastructures9120225

[8] Zhang, P., Gao, Z., Cao, L., Dong, F., Zou, Y., Wang, K., Zhang, Y., & Sun, P. (2022). Marine systems and equipment prognostics and health management: A systematic review from health condition monitoring to maintenance strategy. *Machines, 10*(72). https://doi.org/10.3390/machines10020072

[9] Soltangharaei, V., Ai, L., & Ziehl, P. (2023). Implementation of data-driven approaches for condition assessment of structures and analyzing complex data. In V. Plevris, A. Ahmad, & N. D. Lagaros (Eds.), *Artificial intelligence and machine learning techniques for civil engineering* (pp. 91–119). IGI Global.

[10] Azimi, M., Eslamlou, A. D., & Pekcan, G. (2020). Data-driven structural health monitoring and damage detection through deep learning: State-of-the-art review. *Sensors, 20*(10), 2778. https://www.mdpi.com/1424-8220/20/10/2778

[11] Andrushia, D. A., Anand, N., Walls, R., Paul, D. T., & Arulraj, P. (2023). Automatic detection of surface thermal cracks in structural concrete with numerical correlation analysis. In V. Plevris, A. Ahmad, & N. D. Lagaros (Eds.), *Artificial intelligence and machine learning techniques for civil engineering* (pp. 121–139). IGI Global.

[12] Wang, et al. "Interpretable Domain Knowledge Enhanced Machine Learning Framework on Axial Capacity Prediction of Circular CFST Columns." *Journal of Structural Engineering*, vol. 150, no. 2, 2024, pp. 225-240. MDPI, doi:10.3390/jse15020225.

[13] Tarawneh, A. N., & Saleh, E. F. (2023). Machine Learning Framework for Predicting Failure Mode and Flexural Capacity of FRP-Reinforced Beams. In V. Plevris, A. Ahmad, & N. D. Lagaros (Eds.), *Artificial Intelligence and Machine Learning Techniques for Civil Engineering* (pp. 53-73). IGI Global.

[14] Mansouri, I., Tezcan, J., & Awoyera, P. O. (2023). A Novel Formulation for Estimating Compressive Strength of High-Performance Concrete Using Gene Expression Programming. In V. Plevris, A. Ahmad, & N. D. Lagaros (Eds.), *Artificial Intelligence and Machine Learning Techniques for Civil Engineering* (pp. 75-90). IGI Global.

[15] JRE Editorial Office, et al. "Review of Advanced Road Materials, Structures, Equipment, and Detection Technologies." *Journal of Road Engineering*, vol. 1, no. 1, 2024, pp. 1-42. MDPI, doi:10.3390/jre01010001.

[16] Ampanavos, et al. "Structural Design Recommendations in the Early Design Phase Using Machine Learning." *Journal of Structural Engineering*, vol. 147, no. 12, 2021, pp. 04021178. ASCE, doi:10.1061/(ASCE)ST.1943-541X.0003123.

[17] Wasse, A., Kim, S., & Patel, R. (2024). State-of-the-art review: Seismic design and performance assessment of special concentrically braced frames developed for complex industrial building structures. *International Journal of Steel Structures, 24*(1), 1–25. https://doi.org/10.1007/s13296-024-00815-w

[18] Wang, X., Liu, Y., & Xin, H. (2024). Bond strength prediction of concrete-encased steel structures using hybrid machine learning method. *Structures, 34*, 123–135. https://doi.org/10.1016/j.istruc.2024.01.015

[19] Fu, B., Zhang, L., Chen, Y., & Wang, J. (2023). Dual generative adversarial networks for automated component layout design of steel frame-brace structures. *Automation in Construction, 146*, 104661. https://doi.org/10.1016/j.autcon.2023.104661

[20] Koodiani, H. K., Rahimi, M., & Zhang, T. (2023). Machine learning tools to improve nonlinear modeling parameters of RC columns. *arXiv*. https://arxiv.org/abs/2303.16140

[21] Le, M. V., Prakash, I., & Nguyen, D. D. (2023). Predicting load-deflection of composite concrete bridges using machine learning models. *Journal of Science and Transport Technology, 3*(4), 43–51. https://doi.org/10.58845/JSTT.UTT.2023.EN.3.4.44-52

[22] van der Westhuizen, A. M., Bakas, N., & Markou, G. (2023). Developing an artificial neural network model that predicts the fundamental period of steel structures using a large dataset. In M. Papadrakakis & M. Fragiadakis (Eds.), *Proceedings of the 9th ECCOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering (COMPDYN 2023)* (Athens, Greece, June 12–14, 2023). https://www.researchgate.net/publication/369559428

[23] Pham, A.-D., Ngo, N.-T., & Nguyen, T.-K. (2020). Machine learning for predicting long-term deflections in reinforced concrete flexural structures. *Journal of Computational Design and Engineering, 7*(1), 95–106. https://doi.org/10.1093/jcde/qwaa010

[24] Mustafa, R., Ahmad, M. T., Kumar, A., Kumar, S., Sah, N. K., & Kumar, A. (2024). Prediction of central deflection and slenderness limit for lateral stability of simply supported concrete beam using machine learning techniques. *Asian*

Journal of Civil Engineering, 25, 5443–5466. https://doi.org/10.1007/s42107-024-01122-9

[25] Jin, Y., & Su, H.-J. (2022). Machine learning models for predicting deflection and shape of 2D cantilever beams. *Journal of Manufacturing Science and Engineering, 144*(7), 1–10.

[26] Lai, D., Demartino, C., & Xiao, Y. (2023). Interpretable machine-learning models for maximum displacements of RC beams under impact loading predictions. *Engineering Structures, 281*, 115723.

[27] Nguyen, N.-M., et al. (2023). Early estimation of the long-term deflection of reinforced concrete beams using surrogate models. *Construction and Building Materials, 370*, 130670.

[28] Ababu, E., et al. (2023). Using machine learning algorithms to develop a predictive model for computing the maximum deflection of horizontally curved steel I-beams. *Mathematics, 12*(8), 151.

[29] Sethi, S. (2023). *Optimizing the prediction of small deflections for steel cantilever beams with finite element analysis and deep learning*. Exploratio Journal.

[30] Garza Gutierrez, K., Goble, C., Brooke, J., & Jay, C. (2015). Framing the community data system interface. In *Proceedings of the 2015 British HCI Conference* (pp. 269–270). Association for Computing Machinery.

[31] Ghahramani, Z. (2004). Unsupervised learning. In O. Bousquet, U. von Luxburg, & G. Rätsch (Eds.), *Advanced lectures on machine learning* (pp. 72–112). Springer-Verlag.

[32] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

[33] Polamuri, S. (2023). *10 most popular supervised learning algorithms in machine learning*. Dataaspirant. https://dataaspirant.com/supervised-learning-algorithms

[34] Bartz-Beielstein, T. (2024). Supervised learning: Classification and regression. In *Online machine learning* (pp. 13–22). Springer.

[35] Mahmoud, A. (n.d.). *SteelCantileverBeam*. GitHub. Retrieved December 17, 2024, from https://github.com/AbdulLatifMahmoud/SteelCantileverBeam

[36] Data Professor. (n.d.). *Building the machine learning model* [Infographic]. GitHub. Retrieved December 17, 2024, from https://github.com/dataprofessor/infographic/blob/master/01-Building-the-Machine-Learning-Model.JPG

[37] Data Professor. (2020, April 2). *Machine learning in Python: Building a linear regression model* [Video]. YouTube. Retrieved December 17, 2024, from https://www.youtube.com/watch?v=R15LjD8aCzc

[38] Schaly. (2022, April 20). *Decision tree regression in Python* [Video]. YouTube. Retrieved December 17, 2024, from https://www.youtube.com/watch?v=yvnLpXjrRu8

[39] Educational Research Techniques. (2022, May 22). *Random forest regression with Python* [Video]. YouTube. Retrieved December 17, 2024, from https://www.youtube.com/watch?v=LhBOVWSu-tI

[40] Data Professor. (2020, March 11). *Machine learning in Python: Building a classification model* [Video]. YouTube. Retrieved December 17, 2024, from https://www.youtube.com/watch?v=XmSlFPDjKdc

[41] Hibbeler, R. C. (2017). *Mechanics of materials* (10th ed.). Pearson.

[42] Timoshenko, S. P., & Gere, J. M. (2009). *Theory of elastic stability* (2nd ed.). Dover Publications.

[43] Gere, J. M., & Timoshenko, S. P. (1984). *Mechanics of materials* (3rd ed.). Chapman & Hall.

[44] Mehta, P. K., Monteiro, P. J. M., & McKenna, P. (2006). *Concrete: Microstructure, properties, and materials* (3rd ed.). McGraw-Hill Education.

[45] American Institute of Steel Construction. (2016). *Steel construction manual* (15th ed.). AISC.

[46] ANSYS, Inc. (2024). *ANSYS Mechanical user's guide*.

[47] Autodesk. (2024). *Autodesk Robot Structural Analysis Professional*.

APPENDIX A

Connect python with ETABS

```
#create API helper object
helper = comtypes.client.CreateObject('ETABSv1.Helper')
helper = helper.QueryInterface(comtypes.gen.ETABSv1.cHelper)

try:
    #get the active ETABS object
    myETABSObject = helper.GetObject("CSI.ETABS.API.ETABSObject")
except (OSError, comtypes.COMError):
    print("No running instance of the program found or failed to
attach.")
    sys.exit(-1)

#create SapModel object
mySapModel = myETABSObject.SapModel
model_name = mySapModel.GetModelFilename()
```

Cantilever beam configration in ETABS

```
# Function to create a new model and draw the cantilever
def create_model(mySapModel, length, point_load):
    # Unlock the model to allow modifications
    ret = mySapModel.SetModelIsLocked(False)
    # Delete existing frame objects
    num_frames = 0
    frame_names = []
    [num_frames, frame_names, ret] =
mySapModel.FrameObj.GetNameList(num_frames, frame_names)

    for frame in frame_names:
        ret = mySapModel.FrameObj.Delete(frame)

    ret = mySapModel.SetPresentUnits(6)  # KN-m units

    # Define a cantilever beam (1 point fixed at one end, length
is variable)
    frameName = " "
    [frameName, ret]= mySapModel.FrameObj.AddByCoord(0, 0, 0,
length, 0, 0, frameName, "FrameAutoSelect", "1", "Global")

    # Apply point load at the free end
    # assign point object restraint at base
    PointName1 = " "
```

```
    PointName2 = " "
    Restraint = [True, True, True, True, True, True]
    [PointName1, PointName2, ret] =
mySapModel.FrameObj.GetPoints(frameName, PointName1, PointName2)
    ret = mySapModel.PointObj.SetLoadForce(PointName2, "Dead",
[0, 0, -point_load, 0, 0, 0])
    ret = mySapModel.PointObj.SetRestraint(PointName1, Restraint)
    PointName1 = " "
    PointName2 = " "
    Restraint = [False, False, False, False, False, False]
    [PointName1, PointName2, ret] =
mySapModel.FrameObj.GetPoints(frameName, PointName1, PointName2)
    ret = mySapModel.PointObj.SetRestraint(PointName2, Restraint)
    return frameName
```

Run analysis and design in ETABS

```
# Function to run analysis-Design and retrieve results
def run_analysis_and_save_results(mySapModel, frameName):
    # Run the analysis
    ret = mySapModel.Analyze.RunAnalysis()
    if ret != 0:
        print("Analysis failed.")
        return

    # Auto-design the section
    ret = mySapModel.DesignSteel.StartDesign()
    if ret != 0:
        print("Steel design failed.")
        return
    # Auto-design the section
    ret = mySapModel.DesignSteel.StartDesign()
    if ret != 0:
        print("Steel design failed.")
        return
    # Set up variables for design results retrieval
    NumberItems = 0
    FrameName = []
    FrameType = []
    DesignSect = []
    Status = []
    PMMCombo = []
    PMMRatio = []
    PRatio = []
    MMajRatio = []
    MMinRatio = []
    VMajCombo = []
    VMajRatio = []
    VMinCombo = []
    VMinRatio = []

    # Retrieve the design results for the specified frame
    [NumberItems,
     FrameName,
     FrameType,
     DesignSect,
     Status,
     PMMCombo,
     PMMRatio,
```

```
     PRatio,
     MMajRatio,
     MMinRatio,
     VMajCombo,
     VMajRatio,
     VMinCombo,
     VMinRatio,
     ret] = mySapModel.DesignSteel.GetSummaryResults_3(
        frameName,  # Specific frame name
        NumberItems,
        FrameName,
        FrameType,
        DesignSect,
        Status,
        PMMCombo,
        PMMRatio,
        PRatio,
        MMajRatio,
        MMinRatio,
        VMajCombo,
        VMajRatio,
        VMinCombo,
        VMinRatio,
        0  # eItemType.Objects to specify exact frame
    )

    # Check if data was successfully retrieved
    if ret == 0 and NumberItems > 0:
        # Collect and print the results
        results = []
        for i in range(NumberItems):
            results.append({
                "Design Section": DesignSect[i],
            })

        # Display the results
        #for result in results:
            #print(result)
    else:
        print("Failed to retrieve design results or no results
available.")


    # Retrieve reactions at support
    PointName1 = " "
    PointName2 = " "
    [PointName1, PointName2, ret] =
mySapModel.FrameObj.GetPoints(frameName, PointName1, PointName2)

    # get results for load cases DD
    NumberResults = 0
    Frame_Num = []
    Elm = []
    ACase = []
    StepType = []
    StepNum = []
    U1 = []
    U2 = []
    U3 = []
    R1 = []
    R2 = []
```

```
    R3 = []
    ObjectElm = 0
    reactions = []
    ret =
mySapModel.Results.Setup.DeselectAllCasesAndCombosForOutput()
    ret =
mySapModel.Results.Setup.SetCaseSelectedForOutput('Dead')

    [NumberResults, Frame_Num, Elm, ACase, StepType, StepNum, U1,
U2, U3, R1, R2, R3,
     ret] = mySapModel.Results.JointDispl(PointName2, ObjectElm,
NumberResults, Frame_Num, Elm, ACase, StepType,
                                          StepNum, U1, U2, U3,
R1, R2, R3)
    # Check if data was successfully retrieved
    if ret == 0 and NumberResults > 0:
        # Collect and print the results
        for i in range(NumberResults):
            results.append({
                "U3": U3[i],
            })

        # Display the results
        #for result in results:
            #print(result)
    else:
        print("Failed to retrieve design results or no results
available.")
    return results
```

Export data to Excel

```
# Function to export data to Excel
def export_to_excel(Length, PointLoad, SelectedDesign, U3,
session_id, SteelGrade):
    DesignResults = []
    DesignResults.append([SteelGrade, Length, PointLoad,
SelectedDesign, U3])
    df = pd.DataFrame(DesignResults, columns=["Steel Grade",
"Length", "Point Load", "Selected Design", "U3"])
    if os.path.exists(f'results_session_{session_id}.xlsx'):
        # Save the updated DataFrame back to the same file
        existing_df =
pd.read_excel(f'results_session_{session_id}.xlsx')
        combined_df = pd.concat([existing_df, df],
ignore_index=True)

combined_df.to_excel(f'results_session_{session_id}.xlsx',
index=False)
    else:
        df.to_excel(f'results_session_{session_id}.xlsx',
index=False)
    return
```

Main Function

```
# Main function to run the whole process
def run_etabs_robot(mySapModel,lengths, point_loads,
steel_properties, session_id):
    # Loop over each combination of length and point load
    for SteelGrade in steel_properties:
        get_i_section_properties(mySapModel, SteelGrade)
        for length in lengths:
            for point_load in point_loads:
                X = []
                Y = []
                frameName = create_model(mySapModel,length,
point_load)
                results =
run_analysis_and_save_results(mySapModel, frameName)
                X.append(results[0])
                Y.append(results[1])
                export_to_excel(length, point_load, X, Y,
session_id, SteelGrade)
```

Run the Function in secions

```
#Lists of lengths and point loads
for i in range(1, 6):  # i goes from 1 to 5 (inclusive)
    for j in range(1, 6):  # j goes from 1 to 5 (inclusive)
        session_id = str(i) + str(j)
        point_load_list =
[i+0.1,i+0.2,i+0.3,i+0.4,i+0.5,i+0.6,i+0.7,i+0.8,i+0.9]
        length_list =
[j+0.1,j+0.2,j+0.3,j+0.4,j+0.5,j+0.6,j+0.7,j+0.8,j+0.9]
        # Run the ETABS robot
        run_etabs_robot(mySapModel, length_list, point_load_list,
steel_properties, session_id)
```

APPENDIX B

Decision Tree Classifier

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

# Load the data
df = pd.read_csv('Steel_Section_Design_Transfered.csv')

#data separeation into x and y
Y = df['Selected Design']
X = df[['Steel Grade', 'Length', 'Point Load']]

# Encode the categorical "Steel Grade" column
encoder = LabelEncoder()
X['Steel Grade'] = encoder.fit_transform(X['Steel Grade'])

# spliting the data into 80½ for training the model and 20½ for
testing the model
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=100)

# Building the ML model
clf = DecisionTreeClassifier()
# Traning the model
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)

print(Y_pred)
print(X_test)
print(Y_test)

# Appling model to preedict y
print(clf.score(X_test, Y_test))

# Plotting feature importances
feature_importances = pd.Series(clf.feature_importances_,
index=X.columns)
feature_importances.plot(kind='bar')
plt.xlabel("Feature")
plt.ylabel("Importance")
plt.title("Feature Importance in Decision Tree")
plt.subplots_adjust(bottom= 0.25)  # Adjust left and bottom
margin as needed
plt.show()
```

Decision Tree Classifier Predictions

```
print(Y_pred)
```

['W14X22' 'W10X12' 'W12X14' 'W8X10' 'W10X12' …….. 'W10X19'
'W12X14' 'W12X19' 'W12X14' 'W16X26']

```
print(X_test)
```

	Steel Grade	Length	Point Load
2152	0	4.5	2.2
117	0	2.7	2.4
168	0	3.3	4.3
801	0	2.2	1.9
1637	0	2.2	5.4
...	...	...	...
1987	0	4.4	1.0
1925	0	3.3	3.7
2497	0	3.9	4.4
2589	0	3.3	5.5
2557	0	4.9	5.3

```
print(Y_test)
```

[521 rows x 3 columns]
2152 W14X22
117 W10X12
168 W12X14
801 W8X10
1637 W10X12
...
1987 W10X19
1925 W12X14
2497 W12X19
2589 W12X14
2557 W16X26

```
print(clf.score(X_test, Y_test))
```

Name: Selected Design, Length: 521, dtype: object
0.946257197696737

APPENDIX C

Decision Tree Regressor

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder

# Calling the data
df = pd.read_csv('Steel_Section_Design_Transfered.csv')

# Separating the data into dependent (X) and independent (Y)
X = df[['Steel Grade', 'Length', 'Point Load', 'Selected
Design']]
Y = df['U3']

# Encode the categorical "Steel Grade" column
encoder = LabelEncoder()
X['Steel Grade'] = encoder.fit_transform(X['Steel Grade'])
X['Selected Design'] = encoder.fit_transform(X['Selected
Design'])

# splitting data 80% for traning and 20% for testing the model
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=100)

# Training the model
regressor = DecisionTreeRegressor()
regressor.fit(X_train, Y_train)
Y_pred = regressor.predict(X_test)

# Creating a DataFrame to compare Y_test and Y_pred
comparison_table = pd.DataFrame({
    'Y_test': Y_test.values,  # Convert Y_test to numpy array if
it's a Series
    'Y_pred': Y_pred
}, index=Y_test.index)  # Ensure the same index is used

# Display only the first few rows
print(comparison_table.head())

# Evaluating the model
mse = mean_squared_error(Y_test, Y_pred)
R2 = r2_score(Y_test, Y_pred)
print('MSE: ', mse)
```

```
print('R2: ', R2)

# Data visualization
plt.scatter(Y_test, Y_pred, alpha= 0.3, color='#1f77b4',
label='Yexp Vs. Predicted')
z = np.polyfit(Y_test,Y_pred,1)
Y_poly = np.poly1d(z)
plt.plot(Y_test, Y_poly(Y_test), label='Best Fit Line')
plt.xlabel('The true value of V_dis mm')
plt.ylabel('The predicted value of V_dis mm')
plt.title('Decision Tree Regressor - 2.6k Dataset - test_size=
0.2')
plt.legend()
```

Decision Tree Regressor Predictions

```
# Display only the first few rows
print(comparison_table.head())
```

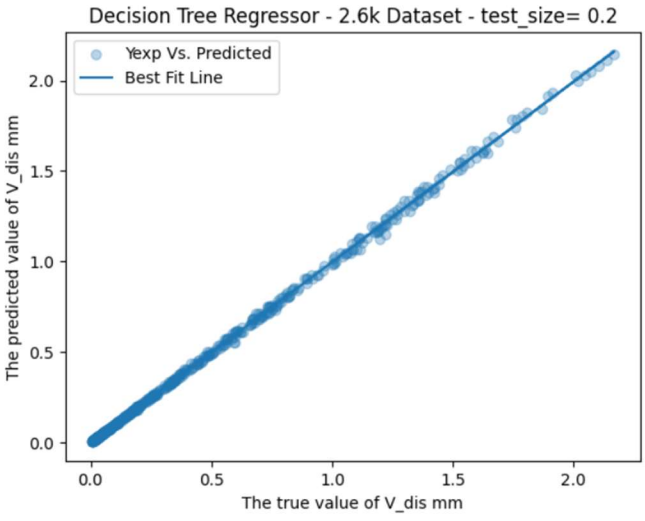
	Y_test	Y_pred
2152	0.533	0.543
117	0.103	0.106
168	0.272	0.276
801	0.049	0.048
1637	0.097	0.095

```
# Evaluating the model
mse = mean_squared_error(Y_test, Y_pred)
R2 = r2_score(Y_test, Y_pred)
print('MSE: ', mse)
print('R2: ', R2)
```

MSE: 0.00015943378119001912

R2: 0.9994104205054629

```
plt.legend()
```



APPENDIX D

Different Regression Comparison

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder

print('Different Regression Comparison - 2.6k Dataset - test
size=0.2')
# Calling the data
df = pd.read_csv('Steel_Section_Design_Transfered.csv')

# Separating the data into dependent (X) and independent (Y)
X = df[['Length', 'Point Load', 'Selected Design']]
Y = df['U3']

# Encode the categorical "Steel Grade" column
encoder = LabelEncoder()
#X['Steel Grade'] = encoder.fit_transform(X['Steel Grade'])
X['Selected Design'] = encoder.fit_transform(X['Selected
Design'])

# splitting data 80% for traning and 20% for testing the model
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=100)

# Traning the model1
reg = LinearRegression()
reg.fit(X_train, Y_train)
# Appling model to preedict y
Y_pred1 = reg.predict(X_test)
# Evaluating the model1
mse_1 = mean_squared_error(Y_test, Y_pred1)
R2_1 = r2_score(Y_test, Y_pred1)
print('MSE_Model1: ', mse_1)
print('R2_Model1: ', R2_1)

# Training the model2
regressor = DecisionTreeRegressor()
regressor.fit(X_train, Y_train)
Y_pred2 = regressor.predict(X_test)
# Evaluating the model2
```

```
mse_2 = mean_squared_error(Y_test, Y_pred2)
R2_2 = r2_score(Y_test, Y_pred2)
print('MSE_Model2: ', mse_2)
print('R2_Model2: ', R2_2)

# Training the model3
clf = RandomForestRegressor()
clf.fit(X_train, Y_train)
Y_pred3 = clf.predict(X_test)
# Evaluating the model2
mse_3 = mean_squared_error(Y_test, Y_pred3)
R2_3 = r2_score(Y_test, Y_pred3)
print('MSE_Model3: ', mse_3)
print('R2_Model3: ', R2_3)

# Creating a DataFrame to compare Y_test and Y_pred
comparison_table = pd.DataFrame({
    'Y_test': Y_test.values,  # Convert Y_test to numpy array if
it's a Series
    'Y_pred1': Y_pred1,
    'Y_pred2': Y_pred2,
    'Y_pred3': Y_pred3
}, index=Y_test.index)  # Ensure the same index is used

# Display only the first few rows
print(comparison_table.head())
# Data visualization

#Model1
polt_1 = plt.scatter(Y_test, Y_pred1, alpha= 0.3,
color='#F6D55C', label='LinearRegression')
z_1 = np.polyfit(Y_test,Y_pred1,1)
Y_poly_1 = np.poly1d(z_1)
plt.plot(Y_test, Y_poly_1(Y_test), color='#F6D55C' , label= 'Best
fit')

#Model2
polt_2 = plt.scatter(Y_test, Y_pred2, alpha= 0.3, color='#3CAEA3'
, label='DecisionTreeRegressor')
z_2 = np.polyfit(Y_test,Y_pred2,1)
Y_poly_2 = np.poly1d(z_2)
plt.plot(Y_test, Y_poly_2(Y_test), color='#3CAEA3', label= 'Best
fit')

#Model3
polt_3 = plt.scatter(Y_test, Y_pred3, alpha= 0.3, color='#ED553B'
, label='RandomForestRegressor')
z_3 = np.polyfit(Y_test,Y_pred3,1)
Y_poly_3 = np.poly1d(z_3)
plt.plot(Y_test, Y_poly_3(Y_test), color='#ED553B', label= 'Best
fit')
plt.xlabel('The true value of V_dis (mm)')
plt.yticks(fontsize=5)  # Adjust the font size as needed
plt.ylabel('The predicted value of V_dis (mm)')
plt.title('Different Regression Comparison - 2.6k Dataset - test
size=0.2')
plt.show()
```

Regression Results

```
print('Different Regression Comparison - 2.6k Dataset - test
size=0.2')
print('MSE_Model1: ', mse_1)
print('R2_Model1: ', R2_1)
print('MSE_Model2: ', mse_2)
print('R2_Model2: ', R2_2)
print('MSE_Model3: ', mse_3)
print('R2_Model3: ', R2_3)
```

Different Regression Comparison - 2.6k Dataset - test size=0.2
MSE_Model1: 0.026088540113477228
R2_Model1: 0.9035256632659217
MSE_Model2: 0.0001535508637236084
R2_Model2: 0.9994321752896772
MSE_Model3: 0.00012331273378118898
R2_Model3: 0.9995439946370835

```
# Display only the first few rows
print(comparison_table.head())
```

	Y_test	Y_pred1	Y_pred2	Y_pred3
2152	0.533	0.746625	0.543	0.53757
117	0.103	-0.092249	0.106	0.10768
168	0.272	0.358330	0.276	0.27212
801	0.049	0.121568	0.048	0.04948
1637	0.097	-0.045939	0.095	0.09239

```
plt.show()
```