EVALUATION OF DATA MINING MODELS FOR PREDICTING CONCRETE STRENGTH

WONG CHUAN MING

UNIVERSITI TUNKU ABDUL RAHMAN

Evaluation of Data Mining Models For Predicting Concrete Strength

Wong Chuan Ming

A project report submitted in partial fulfilment of
the requirements for the award of Bachelor of
Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : *wcm*

Name : WONG CHUAN MING

ID No. : 2106532

Date : 13/9/2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"Evaluation of Data Mining Models For Predicting Concrete Strength"** was prepared by **Wong Chuan Ming** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Software Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr. Khor Kok Chin

Date : 13/9/2024

ACKNOWLEDGEMENTS

First of all, I would like to express gratitude and appreciation to my project supervisor, Dr. Khor Kok Chin, who has provided me with guidance, advice and encouragement throughout the course of this project. With his knowledge, enthusiasm and constructive feedbacks, I am able to improve the standards of my project and accomplish the objectives of this project without trouble.

Furthermore, I would like to also thank Dr. Lim Siong Kang, a civil engineering professor from the Department of Civil Engineering of UTAR. His responsiveness and knowledge in construction materials have helped me understand more about the dataset used in this project which has helped me to improve the quality of my work. I would also like to acknowledge Dr. Wong Whee Yen for her guidance in the first part of the project.

Throughout the course of this project, my friends and family have provided me with encouragement. Their constant encouragements have helped me to boost my confidence and motivation. I sincerely appreciate all of their support throughout these two semesters.

ABSTRACT

Data mining techniques are becoming more popular in recent years due to their abilities to predict any types of data with high accuracy. Conventional techniques for predicting strength of concretes frequently empirical calculations. The efficiency and accuracy of concrete strength prediction can be improved by using data mining techniques. A dataset provided by the Department of Civil Engineering of UTAR that consists of 343 samples of concrete strength along with its mixture proportions is used in this project. The data mining models used in this project are (i) Decision Tree, (ii) AdaBoost, (iii) XGBoost, (iv) Bagging Regressor and (v) Artificial Neural Network. These models are all evaluated with hyperparameter tuning and different feature selection techniques. The feature selection techniques included are (i) Principle Component Analysis, (ii) Boruta and (iii) LASSO. The best performing model is selected and used to generate different sets objective-function that will be selected and used in a Particle Swarm Optimization algorithm to solve a single objective optimization problem that finds the optimal values of each concrete feature to maximize the strength of concrete. The Bagging Regressor model with LASSO is the best performer with a R^2 score of 0.9525. It is selected as the to generate objective-functions for the Particle Swarm Optimization algorithm as it performs performs consistently well with tuned hyperparameters and feature selection. The Particle Swarm Optimization algorithm is able to generate optimal values for the concrete features that maximizes the strength of concrete. The maximum strength that is achievable with the optimal values for each concrete feature found by the Particle Swarm Optimization Algorithm is 27.96.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

y_i	actual observed value
$\hat{y}_i$	predicted value
$\bar{y}_i$	mean of the observed values
n	number of observations
AdaBoost	Adaptive Boosting
XGBoost	Extreme Gradient Boosting
BR	Bagging Regressor
ANN	Artificial Neural Network
DT	Decision Tree
PSO	Particle Swarm Optimization
R^2	Coefficient of Determination
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
v_{id}	Velocity for particle *i* in the d-th dimension
p_{id}	Best previous position for particle *i* in the d-th dimension
x_{id}	The *i*th particle in the d-th dimension
p_{gd}	Global best position found by any particle in the d-th dimension

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

With the recent advancements of the society in recent years, the construction industry plays a crucial role in building the infrastructures around us. In many structures, concrete is the basic building block. Hence, the strength of concretes must be taken into account as it directly influences the strength and durability of these structures. Conventional techniques for estimating the strength of concrete frequently rely on labour- and resource-intensive empirical calculations or comprehensive laboratory testing.

However, in recent years, the use of data mining techniques have become more prevalent as they have become a reliable approach to extract and predict any sort of data. Data mining is the process of searching for patterns in a large batch of unprocessed data and then processing it to extract useful information. In this case, data mining techniques are used to help experts process large amount of data which they can then use to predict whichever data that leads to the most optimal concrete strength. In the construction industry, the accuracy, efficiency and cost-effectiveness of concrete strength prediction could be improved by integrating data mining techniques.

In this project, a dataset related to the strength of lightweight concretes that contain eggshell will be used. This dataset is prepared by the Department of Civil Engineering of UTAR. This dataset will serve as a sample for applying the data mining techniques that will be covered.

1.2 Importance of the Study

This project strives to highlight the capabilities of different data mining models for predicting concrete strength. Data mining models can be used to predict the strength of concretes with high accuracy. As there are many different data

mining models that are used in different types of contexts, it is important to select the one that best fits the current context. Therefore, the data mining models evaluated in this project are compared based on different performance metrics to select the best for predicting concrete strength. The results from this project can provide the construction industry with valuable insights on which data mining models are the most suitable for predicting concrete strength.

Moreover, this project also aims to contribute to academic research by analysing the performances of different data mining models for predicting concrete strength. As different types of optimization techniques are used in this project to predict the strength of concrete optimally, the project can provide insights on the effectiveness of each optimization techniques to researchers.

1.3 Problem Statement

Concrete strength prediction is a crucial task in the construction industry as it affects the safety and durability of structures. Conventional techniques for predicting concrete strength often depend on empirical calculations and laboratory testing. These methods require large amount of time and can take up a lot of resources without providing high accuracy. As datasets grow larger, it is more time consuming to use conventional techniques to manually search through raw data to predict concrete strength. This has caught the attention of many construction professionals, leading them to search for better methods to search through data. Wit hthe development of data mining methods, it is now possible to predict concrete strength more accurately and efficiently without the need of labour. The difficulty lies in choosing the best model to implement for the various and interconnected factors such as aggregate properties, curing conditions and cement content that affect concrete strength.

1.4 Aim and Objectives

1.4.1 Aim

The aim of this study is to assess data mining models for concrete strength prediction, emphasizing how to improve efficiency and accuracy over traditional approaches.

1.4.2 Objectives

i) To train various data mining models for predicting concrete strength
ii) To find the best performing model based on performance metrics
iii) To generate an objective function from the best model and use it in a Particle Swarm Optimization algorithm to solve a single objective optimization problem to maximize the strength of concrete

1.5 Scope and Limitations of the Study

1.5.1 Scope

This project aims to investigate and develop a concrete strength prediction model by using various types of machine learning models such as decision trees and artificial neural networks. The scope includes selecting multiple types of suitable data mining models to be used for evaluation, investigating a comprehensive set of input parameters which is the dataset provided by the Department of Civil Engineering of UTAR, evaluating the performance of the selected data mining models and selecting the best performing model. The best performing model will be used to generate objective-functions that will be used in a Particle Swarm Optimization algorithm to solve a single objective optimization problem to maximize the strength of concrete with an optimal set of feature values.

1.5.2 Limitations

1. Complexity of Model

    ● While a variety of data mining model types will be taken into consideration, not all conceivable algorithms may be covered by the study, and computer resources may limit the complexity of some more complicated models.

2. Practical Implementation Challenges

    ● Integrating data mining models with current processes, obtaining resources, and getting industry acceptability can all present difficulties in their practical application in actual construction projects.

3. Insufficient Training Data

    ● Due to the nature of curing a concrete mixture, the Department of Civil Engineering of UTAR is unable to provide a dataset that is large enough. Hence, the training data used in this project is insufficient.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction to Concrete Formation

For many centuries, the construction industry has relied heavily on concrete, which is a fundamental construction material. It is well known for its adaptability, robustness and durability which have made it a crucial part of many infrastructures throughout the world. Concrete is generally made out of cements, aggregates, water and admixtures. Each of these components affects the concrete's overall performance and mechanical properties. Due to its high density and weight, traditional concrete still presents a number of difficulties in certain applications despite its durability. Researchers and engineers are concentrating on developing alternatives that enhance the properties of materials in response to the expanding need for innovative and sustainable construction methods. Modern concrete is not just a blend of cement, water, and aggregates (Aïtcin, 2000). Minerals, chemical admixtures, fibres, and other additions are commonly added to it. The development of lightweight concrete is one instance of this kind of technological progress.

Lightweight coarse elements such as shale, clay and slate are examples of lightweight coarse elements that are mixed into lightweight concrete so that it has low density. Lightweight concrete is a highly adaptable construction material that has several technological, economic and environmentally beneficial advantages (Haque, Al-Khaiat and Kayali, 2004). It is anticipated to become a popular construction material in the near future. Most lightweight concrete is made with a combination of regular weight sand and lightweight coarse material. The optimal proportions of materials to produce a lightweight concrete are carefully selected to achieve a balance of strength and also reduced weight. The density of lightweight concretes can be decreased if lightweight aggregates are added to the concretes. The materials used in lightweight concretes are dependent on various factors such as the stated strength and

density requirements, thermal conductivity and the price of the aggregates (Thienel, Haller and Beuntner, 2020).

2.2 Concrete Strength

The strength of a concrete directly influences the amount of load a structure is able to withstand. There are many factors that contribute to the overall strength of a concrete such as mixture composition, water to cement ratio and curing conditions. These factors are crucial in order to guarantee that infrastructures are safe and durable. In the construction industry, the durability and reliability of a concrete is determined based on different types of concrete strength. For instance, compressive strength, flexural strength and tensile strength are the most commonly used. Out of all the types of concrete strength, compressive strength is the most popular metric for measuring the strength of concrete. The compressive strength of a concrete measures its ability to endure compressive external forces. Hence, it will be the main output variable that will be used in this research. The dependent variable in this research will therefore be the compressive concrete strength as measured after 28 days of traditional curing.

2.3 Methods for predicting concrete strength

2.3.1 Traditional Methods

In most cases, predicting the strength of concrete by following traditional methods has always depended on empirical approaches which involve lengthy and laborious procedures. In this regard, engineers usually undertake complex research work in the laboratory where they subject concrete samples to different curing conditions and observe closely how their strength changes with time. These procedures involve creating several samples and solidifying them to let them harden over a period of weeks or even months. As such, the empirical approaches necessitate rigorous documentation and analysis to evaluate the impact of parameters such as mix proportions, curing conditions, and environmental variables on the strength of concrete. They are highly laborious processes that are aimed at determining the relationship between composition

of concretes and its strength, hence useful in providing insights for construction projects. However, since empirical techniques are both time-consuming as well as resource intensive there is an emerging interest for investigation into more efficient and technologically advanced ways of measuring concrete strengths in modern day construction practices.

2.3.2 Data Science Methods

Data science methods are more effective ways to predict concrete strength compared to traditional empirical methods. Data scientists uses advanced statistical analysis and machine learning techniques to visualize the complicated relationships between concrete strength and other factors and also the patterns in concrete performance. This approach is a lot faster compared to conventional methods. Machine learning models are able to predict concrete strength accurately when they are trained with real-world data with multiple variables. With this accelerated process, decision-making in construction projects becomes faster by optimizing the allocation of resources and minimizing the requirement for significant laboratory experiments. Additionally, data-driven approaches consistently gathers new knowledge in self-adjusting ways which improves their predictive abilities as they get acquainted with more data. By using data science methods, the process of predicting concrete strength can be simplified and also aligns it with the contemporary construction practices, emphasizing on efficiency and novelty.

An example of a data science method to predict concrete strength is to use machine learning models such as regression models or neural network. Historical datasets that contain several related variables and the resulting strength of concrete can be used to train these models. The models will be able to learn from the datasets provided. After they gain knowledge on the patterns and relations between the variables, the models are then able to predict the strength of concretes based on the related variables on new sets of input data.

For instance, many applications of artificial neural networks (ANN) are created by training the ANN using datasets. ANN is a computational system

that is developed based on the human brain's structure and functionality . It is a network system that is composed of interconnected nodes that help to process information. ANN has proven to be efficient in many fields such as structural engineering, material science and structural analysis and design (Ly, Nguyen and Tran, 2021). The ANN is able to continually refresh its training with new data which allows it to adjust to new data well. The ANN algorithm can be trained with large datasets and also high dimensional datasets. It is able to identify complex relationships and patterns between variables in the data during the training stage so that it can generalize better when new input data are provided. ANN has been proven to be useful when addressing issues related to incomplete or missing data (Lee, 2003).

2.3.3 Summary of Methods

The use of data science methods is much more superior to traditional empirical methods for predicting concrete strength. Traditional methods require labour and may not be effective when dealing with complex datasets. Predicting the mechanical, rheological and durability characteristics of concrete involved deriving empirical relationships from the statistical analysis of experimental data (Ben Chaabene, Flah and Nehdi, 2020). By using linear and non-linear regression models, the correlations between variables in a dataset can be identified easily as these models are capable of understanding and capturing the complex data patterns that human are unable to. Machine learning and data-driven methods are vital in multiple disciplines including spam classifiers, advertising systems, fraud detection and abnormal event detections (Chen et al., 2016). Machine learning models are capable of capturing complex patterns in a dataset without human intervention. These algorithms have the capability to acquire patterns from data through machine learning, rather than relying on direct human programming (Song et al., 2021). These machine learning models have the ability to understand complex relationships between variables when they are provided with input data such as the curing conditions and mix proportions. In contrast, conventional empirical methods require labour to conduct laboratory tests which may take a significantly longer time. Hence,

using machine learning models is more effective and less time-consuming compared to conventional methods for predicting concrete strength.

Based on the analysis of both conventional and data science methods, it is concluded that data science methods outperforms conventional methods as they require less resources and can understand the complex patterns of data. The adaptability and scalability of data science methods enables continuous learning and improvement, which is vital in an industry characterized by ever-changing concepts and environment.

Ultimately, my analysis strongly supports the implementation of data science-driven algorithms for accurately predicting concrete strength. Integrating machine learning algorithms improves prediction accuracy and provides the construction industry with a proactive approach that can adjust to changing needs and conditions. Data mining algorithms play very important roles in the construction industry to ensure that predictions of concrete strength are accurate and infrastructures are safe and durable.

2.4 Existing Works Using Data Science Methods

Based on the review of multiple literatures, ensemble models tend to outperform single models. Adaptive Boosting (AdaBoost), Bagging Regressor (BR) and Extreme Gradient Boosting (XGBoost) have emerged as strong competitors by showing more efficacy compared to individual models. Although ensemble models are more popular in recent years, the ANN, which is a single model is still one of the most widely used model for concrete strength prediction because it can capture the subtle features of concrete strength datasets.

2.5 Overview of Data Science Methods

2.5.1 Ensemble Models

The main categories of algorithms in ensemble learning consist of bagging, boosting and stacking. The most popular algorithm among ensemble models is

the AdaBoost model. This uses the concept of boosting and is used to solve problems in various contexts. Boosting methods are used to reduce both bias and variance in supervised machine learning (Ahmad et al., 2022). Boosting methods can be used effectively with datasets that have class imbalance. It works by minimizing the overall error by introducing additional models that focus on the mistakes made in previous iterations (Carmona, Climent and Momparler, 2019). AdaBoost is a type of ensemble learning model that is capable of identifying cases that have been misclassified due to the disjunct problem (Taherkhani, Cosma and McGinnity, 2020). Disjunct data occurs when instances in a dataset are grouped into separate and distinct clusters. This leads to small and isolated clusters that should still be included in the model training process as they might contain significant information. AdaBoost works by using its first training data to develop a weak learner and subsequently modifying the training data distribution according to the prediction performance for the following iteration of weak learner training (Feng et al., 2020). AdaBoost works by training a series of weak classifiers repeatedly on different parts of training data. In each iteration, the significance of the misclassified samples of the previous iteration is assessed and then increased by assigning them with greater weights. This procedure ensures that the weak classifiers focus more on the samples which they categorized incorrectly. Hence, this leads to an increase in their performances.

The XGBoost is also one of the most popular boosting algorithms that is widely used for different contexts. XGBoost is a type of ensemble learning model that is classification and regression tree-based which means that it is able to deal with classification and regression tasks. Unlike AdaBoost, gradient boosting machines do not include any form of regularization in their implementations which helps it to mitigate overfitting. Regularization techniques such as Lasso and Ridge penalize large weights in a model by shrinking the weights of the variables towards zero. Hence, this facilitates variable selection, an essential part of high-dimensional problems (Carmona, Climent and Momparler, 2019). In recent years, XGBoost has emerged as a popular ensemble algorithm as it is able to deal with different types of data. In tasks that require tabular data, XGBoost is commonly used. XGBoost is a robust

machine learning model as it has two self-compatible regulatory functions, shrinkage and column subsampling (Kang, Yoo and Gupta, 2021). When the size of a dataset is large, XGBoost can efficiently handle the dataset as it employs several types of optimization techniques such as parallelization. Besides that, XGBoost is capable of identifying and dealing with missing values (Dhaliwal, Nahid and Abbas, 2018).

The BR model is also one of the most popular ensemble models in regression tasks. Hence, it is able to perform well in the context of predicting concrete strength. Bagging works by dealing with bias-variance trade-offs and decreasing the variance of a model to avoid overfitting of data (Biswal, 2021). This results in a more consistent model that usually outperforms single models, particularly when the single models show off significant variability (Ahmad et al., 2022). The BR uses a sampling method that is unconventional to substitute the missing data with data from the original dataset. By employing sampling with replacement, it is feasible for certain observations to be replicated in every subsequent training dataset. In a BR model, the probability of each sample being included in the dataset is the same. Unlike other models, the predictive capabilities of the BR model is not affected by the dimensionality of the training set. The ensemble model that is formed by combining the predictions of the various models calculates the average of all the predictions and uses it as a final prediction. In regression, the prediction can be determined through calculating the mean or average of the predictions generated by multiple models (Ahmad et al., 2021).

2.5.2 Single Models

The ANN is a machine learning model that is designed by emulating the human brain. It comprises of interconnected nodes which are referred to as artificial neurons that are arranged in layers. The output of each node is determined with the aid of the locally accessible information at the node, which can both be saved internally or acquired through the weighted connections (Dongare, Kharde and Kachare, 2012). Each unit takes many inputs from numerous nodes and transmits its output to other nodes. ANNs are extensively employed in the

discipline of machine learning to perform many tasks such as classification, regression and pattern recognition. The ANN has the capability to model complex nonlinear relationships, which differentiates it from the conventional regression approach (Zou, Han and So, 2009). It possesses excellent fault tolerance and high speed and scalability because of its ability to do parallel processing. Conventional prediction models had been created with the use of a predetermined equation structure that is based on a small quantity of data and parameters. If the new data extensively deviates from the previous data, the model should also update its coefficients and modify the equation structure. However, ANN does not require a particular equation format. Instead, it only requires enough amount of input-output data. The system has the functionality to constantly update its training with new data which allows it to easily adapt to new data (Lee, 2003).

2.5.3 Summary

All of the models which are discussed above have shown promising results compared to other models for predicting concrete strength based on other literatures. Hence, all the models discussed will be evaluated which are the AdaBoost, XGBoost, BR and ANN. These models will all be trained, tested and then compared to find out which model produces the most optimal results for predicting concrete strength.

2.6 Overview of Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a type of metaheuristic algorithm that is inspired by intelligent swarm behaviours exhibited by some animals such as a school of fish (Wang, Tan and Liu, 2018). Due to its simplicity of implementation and ability to optimize different types of complex problems, the PSO has become more prevalent in recent years (Song and Gu, 2004). Unlike other population-based algorithms such as genetic programming, the PSO algorithm is derived from the simulation of social behaviour (Shi and Eberhart, 1999). In a PSO algorithm, a candidate solution is represented as a particle. Each particle is associated with its own position which represents the current particle,

velocity which represents the rate of position change and best previous position which represents the positions that gives the best fitness value so far. The equations below shows how the particles are manipulated in a PSO algorithm:

$$x_{id} = x_{id} + v\_id \tag{2.1}$$

$$v_{id} = v_{id} + c_1 * rand( ) * (p_{id} - x_{id}) + c_2 * Rand( ) * (p_{gd} - x_{id}) \tag{2.2}$$

where

v_{id} = velocity for particle i in the d-th dimension

c_1 and c_2 = two positive constants

p_{id} = best previous position for particle i in the d-th dimension

x_{id} = the ith particle in the d-th dimension

p_{gd} = global best position found by any particle in the d-th dimension

rand() and Rand() = two random functions in the range [0,1]

Equation (2.1) is the equation of motion for particles to iteratively update their positions when searching for the optimal solution of a problem. In equation (2.2), the goal is to determine how the position of a particle should change in the next iteration based on three criterias:

i) Inertia (Current Velocity)

- The term v_{id} is the current velocity of the particle. It helps to prevent the particle from changing its direction significantly (Marini and Walczak, 2015).

ii) Cognitive Component (Personal Influence)

- The term $c_1 * rand( ) * (p_{id} - x_{id})$ represents the tendency of particles to return to the best positions that they have found previously.

iii) Social Component (Swarm Influence)

- The term $c_2 * Rand( ) * (p_{gd} - x_{id})$ represents the tendency of a particle to be pulled towards the best position globally that is found by the whole swarm (p_{gd})

Equations (2.1) and (2.2) are repeated iteratively until a stopping criterion is met by the algorithm such as a maximum number of iterations since the most recent update of the global best position (p_{gd}).

In PSO, no selection operation is performed on the particle positions. All the particles in PSO are retained throughout the process as candidate solutions of the population (Shi and Eberhart, 1999). The benefit of PSO is its independence from the gradients of optimization problems (Pedersen and Chipperfield, 2010). This allows it to be implemented in many complex optimization problems. This is particularly beneficial when the gradient of a problem is too complex to derive. However, the search complexity of PSO increases significantly when the dimension of search space increases. As the complexity of an application task increases, it is necessary to increase the parameters and their dimensions which leads to an expansion of the solution space (Juneja and Nagar, 2016).

2.7 Overview of Dataset

The experimental data used in this research is prepared by the Department of Civil Engineering of UTAR. The data consists of a total of 343 samples, with 14 input variables and one output variable. The explanation for each of the attributes can be observed in Table 2.1.

Table 2.1: Description of Attributes

Attribute	Description
w/c	Water to cement ratio
ES/C	Eggshell to concrete ratio

IvST dia.	Diameter of a component or parameter related to the concrete mix
ρb mortar density	Density of the mortar portion of the lightweight concrete mix
V foam (theo) % of total volume	Theoretical percentage of foam volume in relation to the total volume of the lightweight concrete mix
mass foam added % (of total mass)	Percentage of foam mass added to the total mass of the lightweight concrete mix
type of mould	Type of mold used for casting the concrete specimens
age	Number of days the concrete is cured
target ρ	Target density of the lightweight concrete
fresh ρ	Density of the fresh (uncured) lightweight concrete mix
HD	Hardened Density
SSD D	Saturated Surface Dry Density
OD D	Oven Dry Density
Strength	Dependent variable representing the compressive strength of the lightweight concrete

Based on the dataset, the Five-number summary of the variables and also the mean and standard deviation are computed. Since there are only two types of values for the type of mould, it will be considered a classifier variable. Hence, 0 represents 3G and 1 represents sm for the type of mould. Table 2.2 depicts the computed values of the variables.

Table 2.2: Computed Values of Variables

Variable	Min	Max	FQ	SQ	TQ	Mean	Std dev
w/c	0.52	0.72	0.56	0.60	0.64	0.61	0.04
ES/C	0.00	0.10	0.025	0.05	0.075	0.05	0.04
IvST dia.	505.00	790.00	571.25	620.00	680.00	630.05	68.36
ρb mortar density	1905.00	2050.00	1950.00	1985.00	2015.00	1981.46	42.70
V foam (theo) % of total volume	26.20	57.90	30.10	40.10	48.10	40.71	9.57
mass foam added % (of total mass)	1.18	4.81	1.41	2.035	2.84	2.31	0.94
type of mould	0.00	1.00	0.00	1.00	1.00	0.70	0.46
age	7.0	28.00	7.0	7.0	28.00	17.10	10.51
target ρ	800.00	1400.00	1000.00	1200.00	1400.00	1151.59	198.27
fresh ρ	810.00	1430.00	1023.00	1216.00	1410.00	1175.64	195.43
HD	797.00	1446.00	1007.25	1185.50	1398.25	1159.18	198.99
SSD D	882.00	1550.00	1127.50	1270.50	1469.75	1274.16	181.38
OD D	647.00	1302.00	856.25	1012.50	1209.75	998.58	181.53
Strength	0.53	8.69	1.475	2.56	5.35	3.34	2.25

Min – Minimum

Max – Maximum

FQ – First Quartile

SQ – Second Quartile

TQ – Third Quartile

Std dev – Standard Deviation

2.8 Performance Metrics

Performance metrics are essential components of regression analysis and to evaluate performance of machine learning models. Plevris et al. (2022) highlighted that a performance metric is a conceptual and mathematical tool used to quantify the degree of accuracy between a predicted outcome and the actual result. The most popular evaluation metrics for regression tasks are the coefficient of determination (R^2), mean squared error (MSE) and root mean squared error (RMSE). In a regression problem, a model is utilized to make predictions for values that are real numbers rather than categorical (Rainio, Teuho and Klén, 2024). In this project, three performance metrics will be used to evaluate the performance of the different machine learning models: R^2, MSE and RMSE.

i) The coefficient of determination is a measure that quantifies the proportion of variability in the dependent variable explained by the variability in the independent variables (Figueiredo Filho, Júnior and Rocha, 2011). The value of R^2 lies within the range of 0 to 1. A score of 1 signifies a perfect match with all data points falling precisely on the regression line, whereas a value of 0 indicates the absence of any relation. When predicting the strength of concrete, which might change because of several circumstances, it is important to determine the extent to which a model can account for this variability. R^2 is not much influenced by the linearity of the regression fitting model (Chicco, Warrens and Jurman, 2021). It is possible to have a very low R^2 value even for a totally linear model. Conversely,

a high R^2 value can occur even when the model is significantly non-linear.

$$R^2 Score = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y}_i)^2} \quad (2.1)$$

where

R^2 Score = coefficient of determination

y_i = actual observed value

$\hat{y}_i$ = predicted value

$\bar{y}_i$ = mean of the observed values

ii) MSE is the average of the squared difference between the actual and predicted values by the model. The unit of the MSE is the square of the measuring unit of the target variable. It is a perfect metric to compare the performance of different models which will be used. In MSE, outliers are penalized more than small errors since they square each term, making it particularly sensitive to outliers. Given that concrete strength is a continuous variable, the MSE quantifies the average difference between the expected and actual strengths. A smaller MSE suggests superior predictive ability as it shows that the average of the model's predictions are closer to the actual values. When dealing with data that follows a normal distribution, MSE is considered an optimal metric for evaluating model performance (Hodson, Over and Foks, 2021). However, it does not provide detailed information about the specific features of model performance that are considered favourable or unfavourable.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2.2)$$

where

MSE = mean squared error

y_i = actual observed value

$\hat{y}_i$ = predicted value

n = number of observations

iii) RMSE is a statistical measure that quantifies the mean vertical deviation between the observed value and the corresponding predicted value on the regression line (Jierula et al., 2021). Simply, it is the square root of MSE. Similar to MSE, RMSE is also sensitive to outliers as it is an extension of MSE. Unlike MSE, it penalizes large errors linearly which may be desirable in certain situations where large errors should be given less weight. The unit of RMSE is the same as the target variable's unit which makes it more interpretable. It can be used to interpret results and report performance of a model.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.3}$$

where

RMSE = root mean squared error

y_i = actual observed value

$\hat{y}_i$ = predicted value

n = number of observations

2.9 Prior Works

In this research, comprehensive review of several literatures is conducted. The associated datasets, significance, uniqueness and methods employed in the literatures are all assessed. The methods employed are various types of machine learning models such as ANN, Gene Expression Programming and Decision Trees. The summarized findings are presented in the table below.

Table 2.3: Articles Proposing Methods for Predicting Concrete Strength

Author	Title	Problem Statement	Dataset	Techniques	Result	Remarks
Lee, S.C. (2003)	Prediction of concrete strength using artificial neural networks	Prediction models based on the maturity concepts require a regression function with one to three parameters. Traditional prediction models have been created using a set equation structure derived from a restricted amount	Experimental training data and test data	Artificial Neural Network (ANN)	The model was able to generate an average determination coefficient, ADC = 0.97 in one of the tests.	

		of data and parameters. If new data significantly deviates from the old data, the model must update both its coefficients and its equation structure.				
Öztaş, A., Pala, M., Özbay, E., Kanca, E., Çagˇlar, N. and Bhatti, M.A. (2006)	Predicting the compressive strength and slump of high strength concrete using neural network	Predicting the compressive strength and slump values of High Strength Concrete (HSC) is challenging due	Various sources provided experimental data. Data were collected for High-Strength Concretes	Neural Network (NN)	The NN model generated an absolute fraction of variance, R^2 = 0.99931, sum of the square error, SSE = 0.004874	

		to its complicated nature. This results in time-consuming and resource-intensive laboratory processes for mix-design, as well as a lack of rational equations in design codes for making such predictions.	(HSCs) including water-to-binder ratio, water content, fine aggregate ratio, fly ash replacement ratio, air-entraining agent ratio, silica fume replacement ratio, and superplasticizer content.		and mean absolute percentage error, MAPE = 1.956208%	
Behnood, A. and Golafshani, E.M. (2020)	Machine learning study of the mechanical	A large number of previously presented models	A complete dataset was gathered from	M5P Model Tree	Coefficient of determination, R^2 = 0.932,	

	properties of concretes containing waste foundry sand	were created for regular concrete and are not as reliable for predicting the mechanical characteristics of other kinds of concrete.	globally public sources, which included information on the mixing proportions and the mechanical property values at different ages.		correlation coefficient R = 0.864	
Farooq, F., Nasir Amin, M., Khan, K., Rehan Sadiq, M., Javed, M.F., Aslam, F. and Alyousef, R. (2020)	A Comparative Study of Random Forest and Genetic Engineering Programming for the Prediction of Compressive	Investigation of machine learning techniques, including gene expression programming (GEP) and ensemble random	Data points attained from published articles	1. Random Forest Regression 2. Genetic Engineering Programming	RF gives an obstinate relation of R^2 = 0.96 whereas GEP gives an obstinate relation of R = 0.90	

	Strength of High Strength Concrete (HSC)	forest (RF) to improve efficiency and accuracy in predicting the mechanical properties of high strength concretes depending on important input parameters				
Feng, D.C., Liu, Z.T., Wang, X.D., Chen, Y., Chang, J.Q., Wei, D.F. and Jiang, Z.M. (2020)	Machine learning-based compressive strength prediction for concrete: An adaptive	It is challenging to obtain an adequate regression expression for this problem because the	1030 concrete compressive strength tests collected in literatures	Adaptive Boosting Algorithm	The algorithm shows a high performance with a coefficient of determination , $R^2 = 0.982$	

	boosting approach	relationship between the compressive strength and the concrete mixture is highly non-linear				
Kaloop, M.R., Kumar, D., Samui, P., Hu, J.W. and Kim, D. (2020)	Compressive strength prediction of high-performance concrete using gradient tree boosting machine	Exploring the efficiency of a multivariate adaptive regression splines (MARS) model for feature extraction to reliably predict the concrete compressive	Hybrid model that combines Multivariate Adaptive Regression Splines Model (MARS) and Gradient Tree Boosting Machine (GBM)	The MARS-GBM model has shown to outperform MARS-KRR and MARS-GPR with a correlation coefficient of R = 0.992		

		strength (CCS) of high-performance concrete (HPC)				
Nguyen, K.T., Nguyen, Q.D., Le, T.A., Shin, J. and Lee, K. (2020)	Analyzing the compressive strength of green fly ash based geopolymer concrete using experiment and machine learning approaches	The relationship between input elements and compressive strength in new types of concrete, such as geopolymer concrete, is intricate and highly nonlinear because of the growing number of input	The dataset comprises 335 samples with 9 input features (fly ash, water glass solution, sodium hydroxide solution, coarse aggregate, fine aggregate, water, concentration of sodium hydroxide	1. Deep Neural Network (DNN) 2. Deep Residual Network (ResNet)	ResNet was the better model for predicting compressive strength of the fly ash based geopolymer concrete with a correlation coefficient, R = 0.9927, mean absolute error, MAE = 0.5536 MPa and root	

		parameters. Conventional approaches are inadequate for forecasting the compressive strength of geopolymer concrete	solution, curing time, curing temperature) and their corresponding compressive strength records obtained from laboratory tests		mean square error, RMSE = 1.2687 MPa	
Nunez, I., Marani, A. and Nehdi, M.L. (2020)	Mixture Optimization of Recycled Aggregate Concrete Using Hybrid Machine Learning Model	Due to the unpredictability of recycled aggregates and the imprecise estimation of its compressive strength, complex methods are	1134 RAC mixture design examples collected from 55 peer-reviewed publications	1. Gaussian Process Model (GP) 2. Gated Recurrent Unit (GRU) 3. Gradient Boosting Decision Trees (GBRT)	GBRT showed the highest predictive accuracy with a coefficient of determination, $R^2 = 0.919$	

		needed for the optimization of the recycled aggregate concrete mixture (RAC)				
Shahmansouri, A.A., Bengar, H.A. and Ghanbari, S. (2020)	Compressive strength prediction of eco-efficient GGBS-based geopolymer concrete using GEP method	Developing an ideal mix design and predicting concrete's compressive strength through experimentation is expensive and time-consuming	A large database was created containing the compressive strength data of ground granulated blast-furnace slag based geopolymer concrete. It includes 351	Gene Expression Programming (GEP)	The best coefficient of determination generated in the validation stages through 20 GEP models is R^2 = 0.940	

			specimens from 117 unique combinations. The modelling input parameters evaluated were specimen age, sodium hydroxide (NaOH) solution concentration, natural zeolite (NZ), silica fume (SF), and ground granulated blast-furnace slag content			

Ahmad, A., Farooq, F., Niewiadomski, P., Ostrowski, K., Akbar, A., Aslam, F. and Alyousef, R. (2021)	Prediction of Compressive Strength of Fly Ash Based Concrete Using Individual and Ensemble Algorithm	Machine learning methods have demonstrated potential in predicting tangible characteristics. It is crucial to conduct a comparative examination of individual algorithms and ensemble methods, namely bagging.	The data utilised in preparing models to predict the strength of concrete were sourced from published literature and are outlined in Appendix A of the study	1. Decision Tree (DT) 2. Ensemble Bagging Approach 3. Gene Expression Programming (GEP)	An ensemble model utilizing a decision tree demonstrates superior performance when compared to an individual decision tree and gene expression programming. The correlation coefficient R2 = 0.911 is documented for DT with bagging	

Ahmad, W., Ahmad, A., Ostrowski, K.A., Aslam, F., Joyklad, P. and Zajdel, P. (2021)	Application of Advanced Machine Learning Approaches to Predict the Compressive Strength of Concrete Containing Supplementary Cementitious Materials	The traditional method of figuring out concrete's mechanical characteristics, especially its compressive strength, requires casting and testing examples, which takes time and resources	1030 data points of concrete using fly ash and blast furnace slag were collected from various sources. Eight elements were utilised as inputs for the models: cement, fly ash, blast furnace slag, water, superplasticizer, coarse aggregate, fine aggregate, and age. Compressive	1. Bagging Model 2. Gene Expression Programming (GEP) 3. Adaptive Boosting (AdaBoost) 4. Decision Tree Algorithms (DT)	Bagging model generated the highest coefficient correlation, R^2 = 0.92	

			strength was the sole output variable			
Asteris, P.G., Skentou, A.D., Bardhan, A., Samui, P. and Pilakoutas, K. (2021)	Predicting concrete compressive strength using hybrid ensembling of surrogate machine learning models	The uncertainty of whether hybrid soft computing models that are derived from conventional soft computing techniques are more effective	Experimental database which consists of 1030 records are compiled from the machine learning repository of the University of California, Irvine	1. Artificial Neural Network (ANN) 2. Linear and Non-Linear Multivariate Adaptive Regression Splines (MARS-L and MARS-C) 3. Gaussian Process Regression (GPR) 4. Minimax Probability Machine	The HENSM model generated a coefficient of determination, R^2 of 0.8894, which is the highest.	

				Regression (MPMR) 5. Hybrid Ensemble Model (HENSM) which consists of five conventional machine learning models and ANN		
Kang, M.C., Yoo, D.Y. and Gupta, R. (2021)	Machine learning-based prediction for compressive and flexural strengths of steel fiber-reinforced concrete	Development of techniques for predicting steel fiber-reinforced concrete is still in its early stages due to limited availability of	220 sets of compressive and flexural strengths data from steel fiber-reinforced concrete which are collected from 22 references	1. Linear Regression 2. Lasso Regressor 3. Ridge Regressor 4. K-nearest Neighbour (KNN) Regressor	XGBoost Regressor showed the best performance with a root mean square error, RSME = 3.6144 and mean	

		data and complexity		5. Decision Tree Regressor 6. Random Forest Regressor 7. AdaBoost Regressor 8. Gradient Boost Regressor 9. XGBoost Regressor	absolute error, MAE = 2.3540	
Ly, H.B., Nguyen, T.A. and Tran, V.Q. (2021)	Development of deep neural network model to predict the compressive strength of rubber concrete	Limitation of empirical approaches as experiments are conducted according to	Rubber concrete database that is constructed by the researchers	Deep Neural Network Model (DNN)	Correlation coefficient, R=0.9874	

		different standards				
Mohammed, A., Rafiq, S., Sihag, P., Kurda, R. and Mahmood, W. (2021)	Soft computing techniques: Systematic multiscale models to predict the compressive strength of HVFA concrete based on mix proportions and curing times	The construction industry's increasing use of high-performance engineered materials, specifically concrete mixtures with high-volume fly ash (HVFA) for LEED compliance, poses challenges	A wide experimental data (a total of 450 tested HVFA concrete mixes) from different academic research studies	1.Linear and non-linear regression 2.Multi-logistic regression 3.M5P-tree 4.Artificial Neural Network (ANN)	M5P-tree generated the highest correlation coefficient, R = 0.8231	

		in predicting the compressive strength				
Moradi, M.J., Khaleghi, M., Salimi, J., Farhangi, V. and Ramezanianpour, A.M. (2021)	Predicting the compressive strength of concrete containing metakaolin with different properties using ANN	The need of precise early-stage prediction of concrete strength and evaluates how certain attributes of extra cementitious materials affect mechanical properties. This research aids in refining concrete mixtures in the	Comprehensive database including 105 and 134 distinctive experimental records of the compressive strength of concretes containing metakaolin was collected from the literature	Artificial Neural Network (ANN)	The ANN showed superior performance with a correlation coefficient of R = 0.9821 compared to empirical approaches with R = 0.6196	

		construction sector.				
Nguyen, H., Vu, T., Vo, T.P. and Thai, H.T. (2021)	Efficient machine learning models for prediction of concrete strengths	Given the complex and nonlinear relationship between concrete constituents and strength outputs, the difficulty in predicting the engineering properties of concrete, especially high-performance concrete, stems	Two popular datasets of compressive and tensile strengths of high performance concrete	1. Support Vector Regression (SVR) 2. Multilayer Perceptron (MLP) 3. Gradient Boosting Regressor (GBR) 4. Extreme Gradient Boosting (XGBoost)	XGBoost is the best performing model with a correlation coefficient, R = 0.98	

		from its nonhomogeneous composition. This makes it necessary to develop reliable predictive models early on in order to minimize the costs and efforts associated with extensive experimentation and input parameter exploration.				

Song, H., Ahmad, A., Farooq, F., Ostrowski, K.A., Maślak, M., Czarnecki, S. and Aslam, F. (2021)	Predicting the compressive strength of concrete with fly ash admixture using machine learning algorithms	Application of machine learning techniques to predict the compressive strength of concrete which contains fly ash	98 data points collected from experimental approach	1. Gene Expression Programming (GEP) 2. Artificial Neural Network (ANN) 3. Decision Trees (DT) 4. Bagging Regressor (BR)	BR generated the highest coefficient relation, R^2 = 0.95. It also confirms its high accuracy by generating values of Mean Absolute Error, MAE = 3.69Mpa, Mean Square Error, MSE = 24.76 and Root mean square error, RMSE = 4.97	

Ahmad, A., Ahmad, W., Aslam, F. and Joyklad, P. (2022)	Compressive strength prediction of fly ash-based geopolymer concrete via advanced machine learning techniques	The increasing environmental effects of cement production have led to the use of substitute materials, such as fly ash-based geopolymer concrete. This type of concrete's strength is hard to predict. Hence, supervised machine learning algorithms are used to improve the prediction	154 data points retrieved from several literatures	1. Decision Tree (DT) 2. Bagging Regressor (BR) 3. AdaBoost Regressor (AR)	The BR model was the most effective in predicting the results with a coefficient of determination, $R^2 = 0.97$	

		accuracy of this concrete.				
Tran, V.Q., Dang, V.Q. and Ho, L.S. (2022)	Evaluating compressive strength of concrete made with recycled concrete aggregates using machine learning approach	Compression test requires huge amount of materials as well as consumes cost and time	721 database samples of experimental results collected from 67 literatures	1. Gradient Boosting (GB) 2. Extreme Gradient Boosting (XGB) 3. Support Vector Machine Regression (SVR) 4. Partial Dependence Plots (PDP) 5. Particle Swarm Optimization (PSO)	GB-PSO shows the highest prediction accuracy with correlation coefficient, R = 0.9356, root mean square error, RMSE = 5.5604MPa, mean absolute error, MAE = 4.2882MPa	

				6. Hybrid Machine Learning Models (GB-PSO, XGB-PSO, SVR-PSO)		

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This project has implemented a systematic workflow that is adopted from the Cross Industry Standard Process for Data Mining (CRISP-DM). This methodology offers a standardised framework for approaching data mining project. The workflow of this research and the process flow of CRISP-DM is depicted in Figure 3.1 and Figure 3.2.

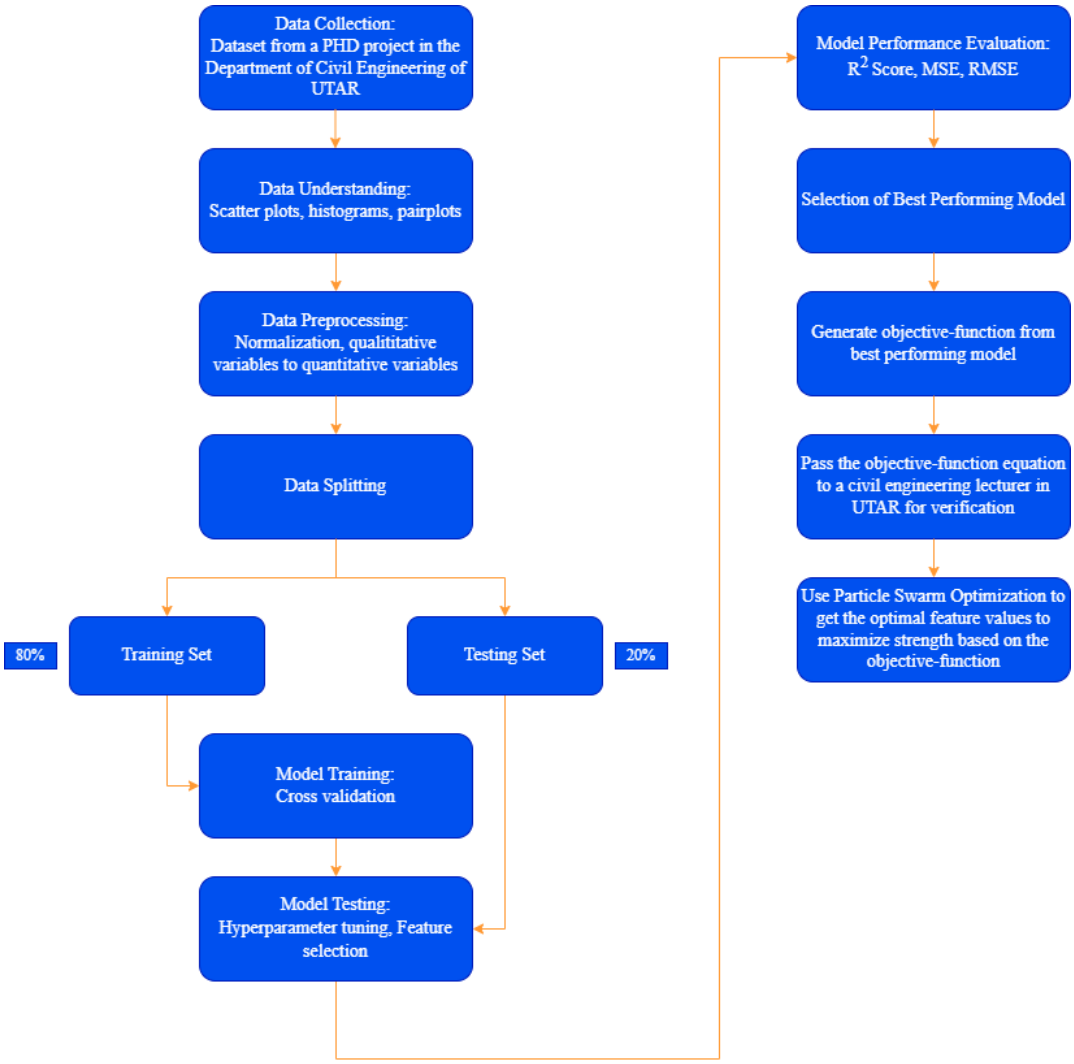

Figure 3.1: Workflow of Research

Figure 3.2: Process flow of CRISP-DM

3.2 Summary of Workflow

The following processes are executed in this research: (i) data understanding, (ii) data preprocessing, (iii) data splitting, (iv) model training, (v) model testing, (vi) model performance evaluation, (vii) selection of best performing model, (vii) definition of objective-functions with best performing model, (viii) verification of objective-functions and (ix) solve single objective optimization problem to maximize strength of concrete.

3.3 Detailed Workflow

3.3.1 Data Understanding

The dataset provided by the Department of Civil Engineering of UTAR has already been prepared well for analysis with data mining models. Before proceeding with data preprocessing, the dataset provided must be visualized and understood well.

Firstly, the shape of the dataset is observed to check the number of rows and features in the dataset. Based on the results, the dataset consists of 343 rows and 15 columns.

```
print(data.shape)
```

```
(343, 15)
```

Figure 3.3: Shape of Dataset

The first five rows from the dataset are printed out in order to gain a better understanding of the data. Based on the five rows printed out, the column "Sample" is the index of the rows and does not provide any predictive capabilities. Hence, it is dropped from the dataset. The sample output of the first five rows are shown in Figure 3.4.

```
print(data.head())
```

```
   Sample   w/c   ES/C  IvST dia.  pb mortar density  \
0       1  0.56  0.00%        560             2040.0
1       2  0.56  0.00%        560             2040.0
2       3  0.56  0.00%        560             2040.0
3       4  0.60  0.00%        575             1995.0
4       5  0.60  0.00%        575             1995.0

   V foam (theo) % of total volume  mass foam added % (of total mass)  \
0                             48.9                                2.84
1                             48.9                                2.84
2                             48.9                                2.84
3                             48.2                                2.66
4                             48.2                                2.66

  type of mould  Age  target r  fresh r    HD  SSD D   OD D  Strength
0            sm    7      1000     1042  1017   1166  879.0      1.10
1            sm    7      1000     1042  1034   1181  895.0      1.08
2            sm    7      1000     1042  1002   1117  867.0      1.35
3            sm    7      1000     1034  1009   1117  858.0      1.29
4            sm    7      1000     1034  1012   1139  866.0      1.39
```

Figure 3.4: Output of first five rows of dataset

After dropping the "Sample" column from the dataset, a concise summary of the dataset is observed using the info() method provided by the pandas library.

```
data=data.drop('Sample',axis=1)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 343 entries, 0 to 342
Data columns (total 14 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   w/c                                   343 non-null    float64
 1   ES/C                                  343 non-null    object
 2   IvST dia.                             343 non-null    int64
 3   pb mortar density                     337 non-null    float64
 4   V foam (theo) % of total volume       337 non-null    float64
 5   mass foam added % (of total mass)     343 non-null    float64
 6   type of mould                         343 non-null    object
 7   Age                                   343 non-null    int64
 8   target r                              343 non-null    int64
 9   fresh r                               343 non-null    int64
 10  HD                                    343 non-null    int64
 11  SSD D                                 343 non-null    int64
 12  OD D                                  328 non-null    float64
 13  Strength                              339 non-null    float64
dtypes: float64(6), int64(6), object(2)
memory usage: 37.6+ KB
```

Figure 3.5: Summary of Dataset

The number of unique values of the type of mould is observed as it is the only categorical variable in the dataset. It is observed that there are only two unique values for the type of mould which are sm and 3G as shown in Figure 3.6.

```
print(data['type of mould'].value_counts())
```

```
type of mould
sm    237
3G    106
Name: count, dtype: int64
```

Figure 3.6: Unique values for type of mould

The number of duplicate rows and null values of the dataset are observed. There is a total of 14 duplicate rows and 31 null values observed from the dataset. Figure 3.7 shows the results of the detected duplicate rows and null values.

```
duplicate_rows = data[data.duplicated()]
print("Number of duplicate rows: ", duplicate_rows.shape)
print("Null values: ")
print(data.isnull().sum())
```

```
Number of duplicate rows:  (0, 14)
Null values:
w/c                                    0
ES/C                                   0
IvST dia.                              0
ρb mortar density                      6
V foam (theo) % of total volume        6
mass foam added % (of total mass)      0
type of mould                          0
Age                                    0
target r                               0
fresh r                                0
HD                                     0
SSD D                                  0
OD D                                  15
Strength                               4
dtype: int64
```

Figure 3.7: Results of detected duplicate rows and null values

A correlation matrix is plotted to gain understanding of the relationship between the dependent variable and the independent variables. This helps to identify how strongly each independent variable is related to the dependent variable. Figure 3.8 shows the correlation matrix of the dataset.

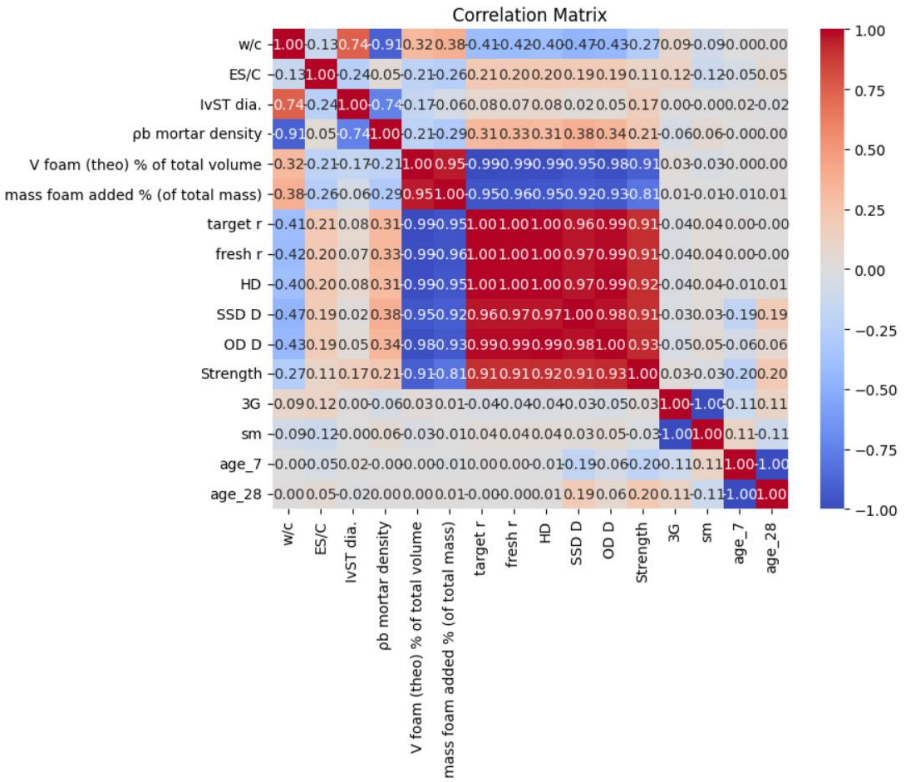

Figure 3.8: Correlation Matrix of the dataset

The highly linear relationships between the independent variables and the dependent variable are visualized using scatter plots. It is seen that the independent variable V foam (theo) % of total volume and the mass foam added % (of total mass) have negative linear relationship with the strength of concrete which means that the concrete strength increases when V foam (theo) % of total volume decreases and vice versa. The scatter plot of the strength of concrete against V foam (theo) % of total volume can be seen in Figure 3.9. The scatter plot of the strength of concrete against the mass foam added % (of total mass) is shown in Figure 3.10.

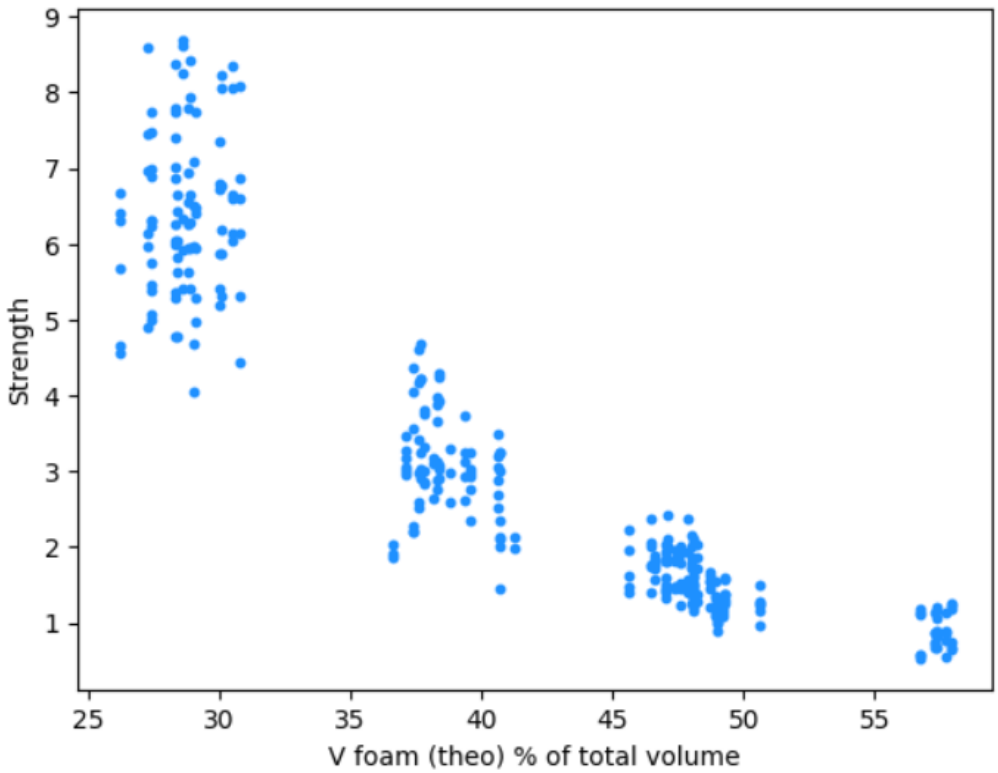

Figure 3.9: Scatter Plot of Strength vs V foam (theo) % of volume

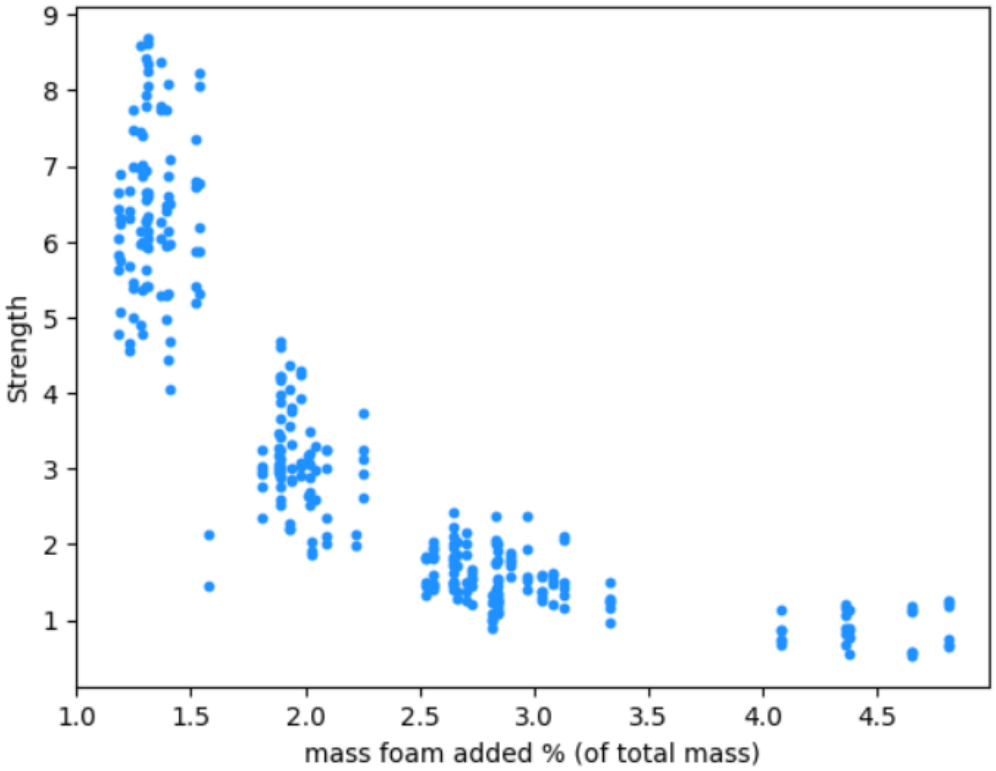

Figure 3.10: Scatter Plot of Strength vs mass of foam added % (of total mass)

The independent variables target r, fresh r, HD, SSD D and OD D have positive linear relationship with concrete strength which means that if they increase, strength of concrete increases as well and vice versa. The scatter plots of each of these independent variables which have high positive linear relationship with concrete strength are shown in the figures below.

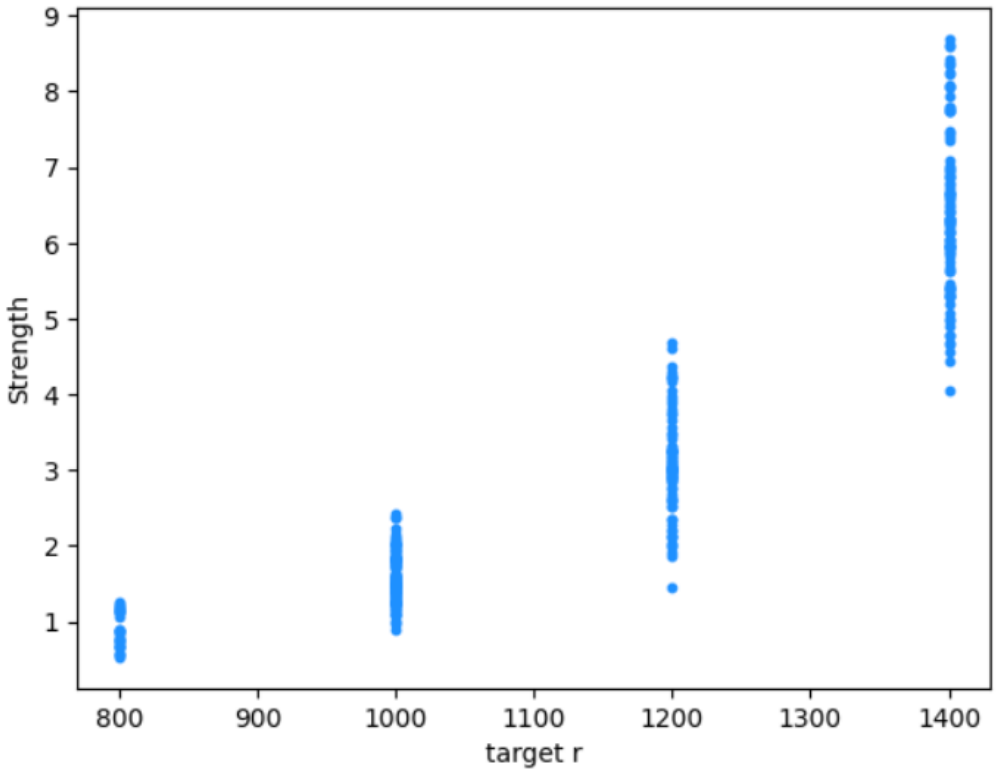

Figure 3.11: Scatter Plot of Strength vs target r

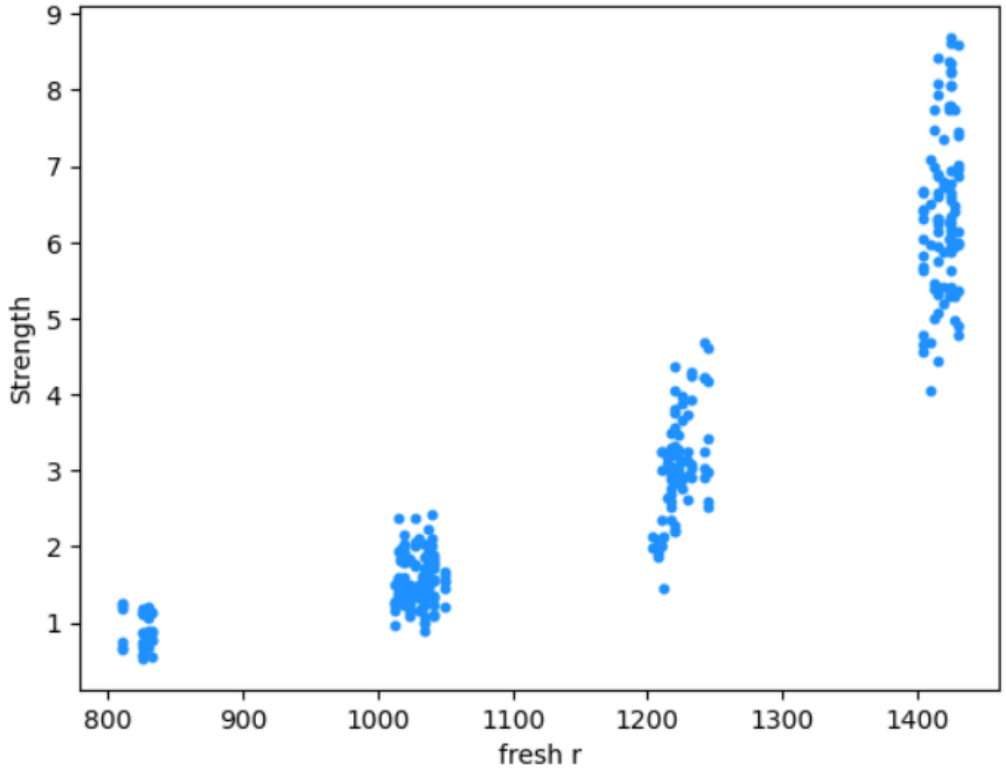

Figure 3.12: Scatter Plot of Strength vs fresh r

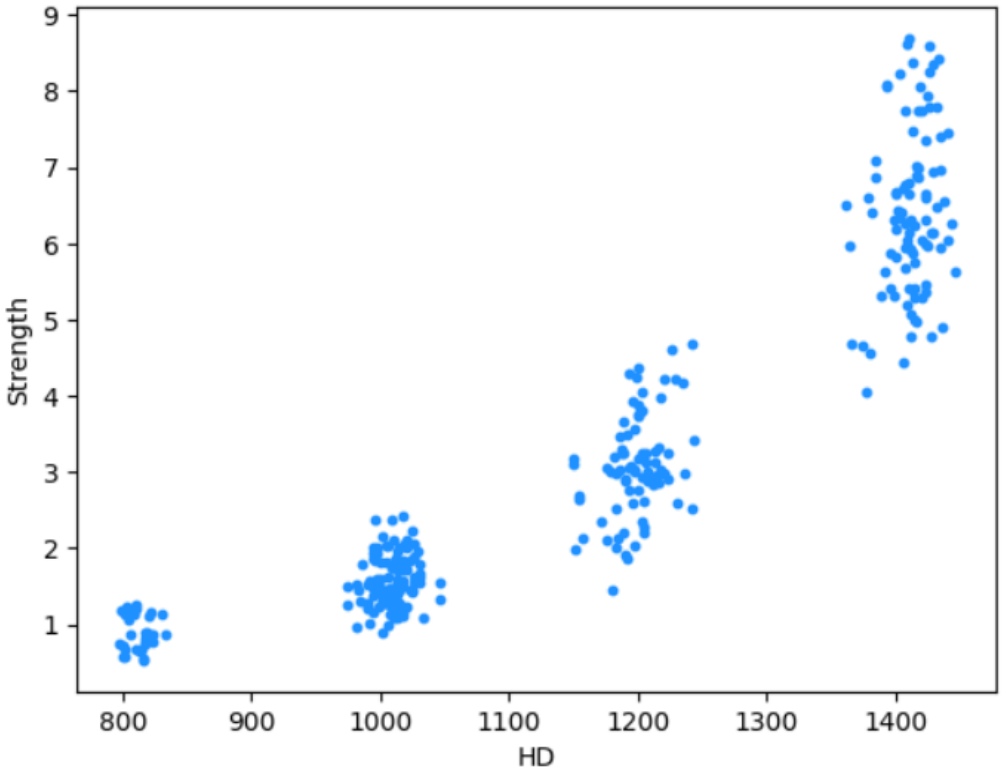

Figure 3.13: Scatter Plot of Strength vs HD

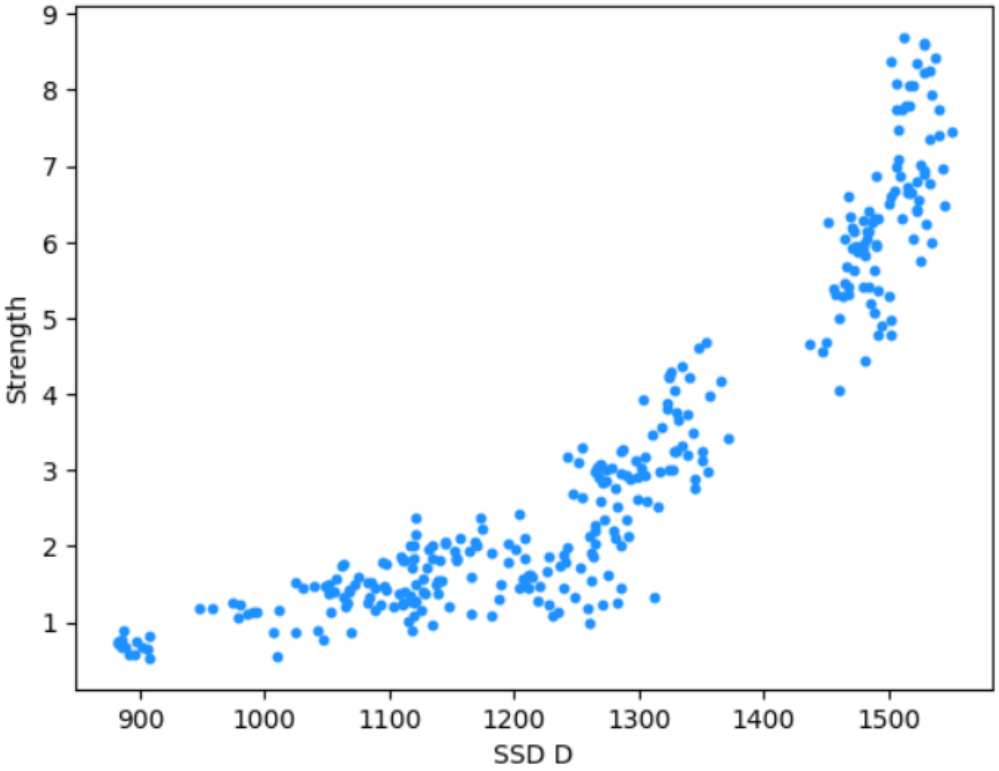

Figure 3.14: Scatter Plot of Strength vs SSD D

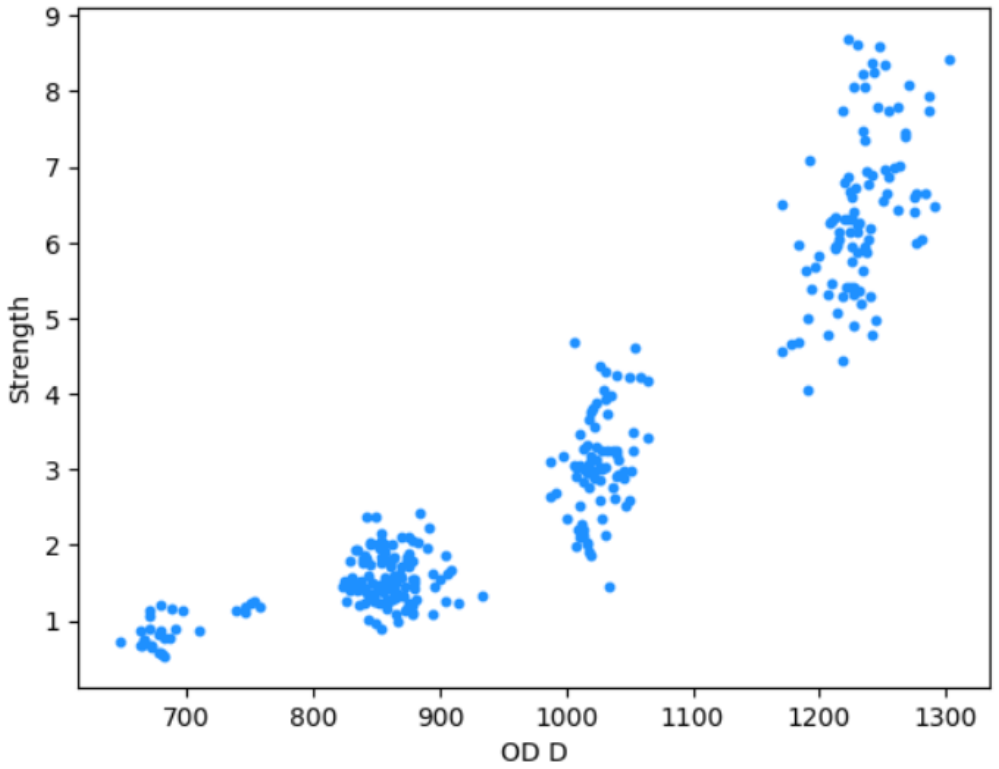

Figure 3.15: Scatter Plot of Strength vs OD D

A bar plot of the categorical variable in the dataset, type of mould, is plotted to visualize the frequency of each value and to check if there is class imbalance. Based on the bar plot in Figure 3.16, the frequency for sm type of mould is higher than 3G. However, the bar plot does not show presence of extreme class imbalance.

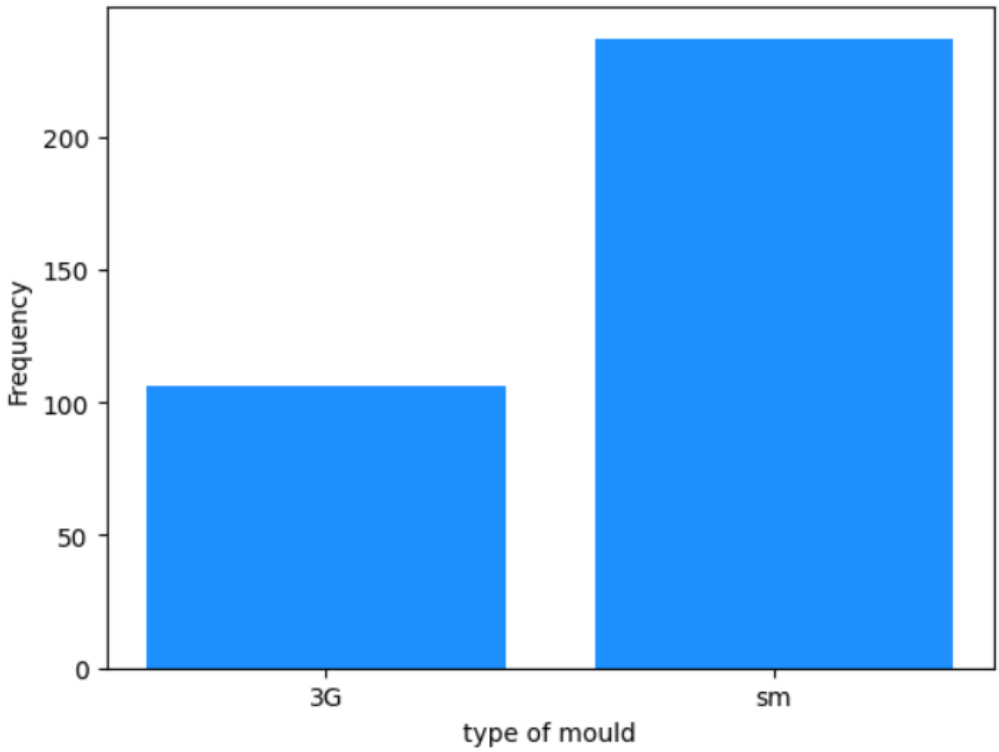

Figure 3.16: Bar Plot of type of mould

Based on domain understanding, values for the variable age are numerical but can only be 7 or 28. Hence, a bar plot is used instead of a histogram to check for imbalance. Figure 3.17 shows the bar plot of the variable age. It is seen that there is no class imbalance present.

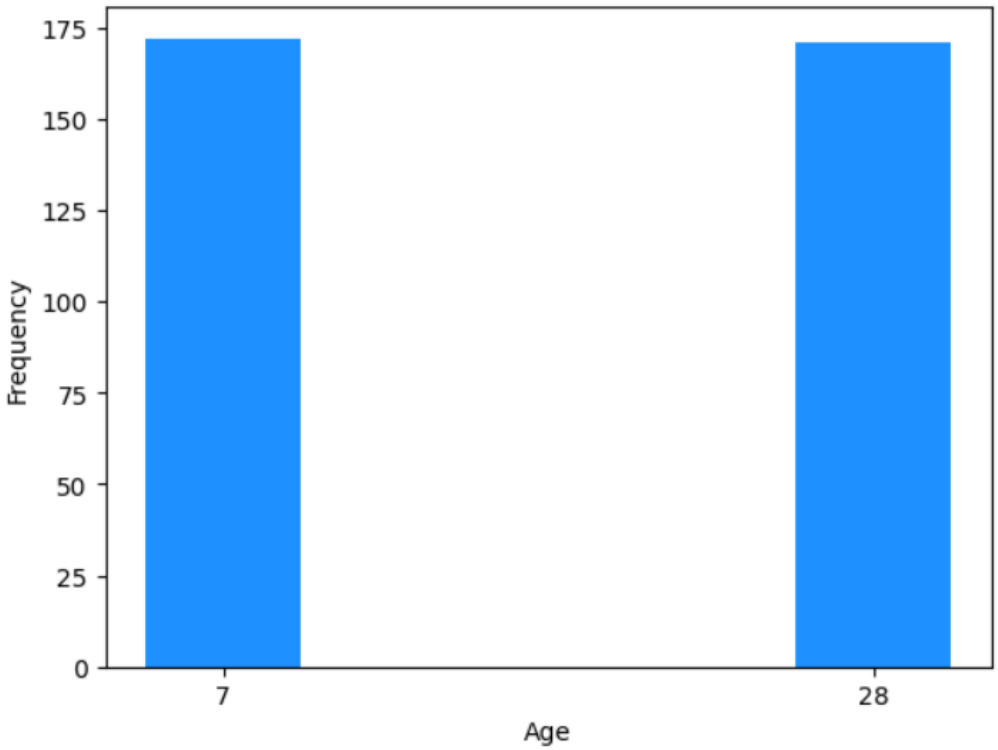

Figure 3.17: Bar Plot of Age

3.3.2 Data Preprocessing

As observed during the data understanding stage, there are null values present in the dataset. Hence, the rows that contain any null value can be dropped completely as the size of the dataset is large. After dropping the rows with null value, there are 318 rows left as shown in Figure 3.18.

```
data = data.dropna()
print(data.shape)

(318, 16)
```

Figure 3.18: Shape of data after removing rows with null value

Based on domain understanding, the independent variable age can only be 7 or 28. Hence, despite it being a numerical variable, it is treated as a categorical variable. The categorical variables in the dataset, type of mould and age, is transformed into numerical formats as some data mining models are not

able to deal with categorical variables. OneHotEncoder is a preprocessing technique that is usually used in machine learning to convert categorical variables into numerical variables. By using the OneHotEncoder, the columns type of mould and age are dropped and replaced by four new columns which are “sm”, “3G”, “age_7” and “age_28”. These columns represent the unique values of each of the categorical variable column. A value of 1 in column “sm” represents that the type of mould is “sm” and vice versa with the same concept being applied to the column “3G”. A value of 1 in column “age_7” means that the age is 7 and a value of 0 means that the age is not 7. The same concept is applied to the column “age_28”. Hence, “age_7” is 0 when “age_28” is 1 and vice versa.

```
encoder = OneHotEncoder()
type_of_mould_reshaped = data['type of mould'].values.reshape(-1, 1)
one_hot_encoded = encoder.fit_transform(type_of_mould_reshaped).toarray().astype(int)
categories = encoder.categories_[0]
new_column_names = [f"{category}" for category in categories]
data[new_column_names] = one_hot_encoded
data.drop(columns=['type of mould'], inplace=True)

age_reshaped = data['Age'].values.reshape(-1, 1)
one_hot_encoded_age = encoder.fit_transform(age_reshaped).toarray().astype(int)
categories_age = encoder.categories_[0]
new_column_names_age = [f"age_{category}" for category in categories_age]
data[new_column_names_age] = one_hot_encoded_age
data.drop(columns=['Age'], inplace=True)
```

Figure 3.19: Code snippet of using OneHotEncoder

Before performing outlier detection and calculations for outlier removal, it is important to ensure that columns containing percentages are properly formatted as numerical values. Hence, the column “ES/C” is formatted to numerical values.

```
data['ES/C'] = data['ES/C'].str.rstrip('%').astype(float) / 100.0
print(data['ES/C'])
```

```
0      0.000
1      0.000
2      0.000
3      0.000
4      0.000
       ...
335    0.025
336    0.025
337    0.025
338    0.025
339    0.025
Name: ES/C, Length: 318, dtype: float64
```

Figure 3.20: Sample rows of ES/C after formatting

The interquartile range (IQR) method and the z-score method are used to detect the presence of outliers in the data. Based on the results, there is no outlier present in the data. Hence, the outlier removal process is skipped in this stage.

```
def detect_outliers_iqr(df, features):
    outliers = pd.DataFrame()
    for feature in features:
        Q1 = df[feature].quantile(0.25)
        Q3 = df[feature].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        feature_outliers = df[(df[feature] < lower_bound) | (df[feature] > upper_bound)]
        outliers = pd.concat([outliers, feature_outliers])
    return outliers.drop_duplicates()
def detect_outliers_zscore(df, features, threshold=3):
    outliers = pd.DataFrame()
    for feature in features:
        z_scores = np.abs((df[feature] - df[feature].mean()) / df[feature].std())
        feature_outliers = df[z_scores > threshold]
        outliers = pd.concat([outliers, feature_outliers])
    return outliers.drop_duplicates()
features = ['w/c', 'ES/C', 'IvST dia.', 'ρb mortar density', 'V foam (theo) % of total volume',
            'mass foam added % (of total mass)', 'Age', 'target r', 'fresh r', 'HD',
            'SSD D', 'OD D', 'Strength', '3G', 'sm']
outliers_iqr = detect_outliers_iqr(data, features)
outliers_zscore = detect_outliers_zscore(data, features)
if len(outliers_iqr) == 0 and len(outliers_zscore) == 0:
    print("No outliers detected")
else:
    total_outliers = len(outliers_iqr) + len(outliers_zscore)
    print(f"Number of outliers detected: {total_outliers}")
    print("Outliers detected using IQR:")
    print(outliers_iqr)
    print("Outliers detected using Z-score:")
    print(outliers_zscore)
```

```
No outliers detected
```

Figure 3.21: Code snippet of outlier detection

The duplicate rows in the dataset are also dropped to prevent bias in analysis. After removing the duplicate rows, it is observed that the number of rows remain the same at 318 rows. This is due to the removal of rows with null values. The rows that contain null values are also duplicates which leads to them being dropped as shown in Figure 3.22.

```
data = data.drop_duplicates()
print("Data shape after dropping duplicate rows: ", data.shape)
```

```
Data shape after dropping duplicate rows:  (318, 16)
```

Figure 3.22: Result of dropping duplicate rows

The independent variables are scaled using StandardScaler so that the range of the features are standardized. This prevents biased results that occur due to features with larger scales being given more weight. The StandardScaler scales the independent variables in the training set and test set by using standardization to prevent higher weightages being given to variables with higher magnitudes and produce bias towards certain variables. Data standardization is the process of rescaling data values to achieve a mean of zero and a standard deviation of one (Jaadi, 2019). The code snippet of scaling the features are shown in Figure 3.23.

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

Figure 3.23: Code snippet of scaling features

3.3.3 Data Splitting

The independent variables are assigned as “x” whereas the target variable is assigned as “y” before splitting. To split the dataset into training set and test set, train_test_split is performed with a test size of 0.2. This means that 80% of the

data will be used to train the models and 20% of the data will be used for testing the models.

```
x = data.drop('Strength', axis=1)
y = data['Strength']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Figure 3.24: Code snippet of data splitting

3.3.4 Model Training

The preprocessed training set and test set are passed onto five machine learning models. Decision Tree, AdaBoost, XGBoost, BR and ANN are the models used in this project. Several different optimization techniques will be applied to each of the models such as feature selection and hyperparameter tuning. This is to evaluate the performance of the models with different optimization techniques. Hence, there will be three categories of optimization techniques for each model: model training without optimization techniques, model training with best hyperparameters found using GridSearchCV and model training with different feature selection techniques. Before starting the experiments, k-fold cross-validation will be used to validate the model to get better estimations on the models' predictive capabilities. The models created will be trained with k-fold cross-validation. In this project, 5-fold cross-validation is used so that there is sufficient amount of data for training while also ensuring that the testing set is large enough to provide reliable estimates of performance.

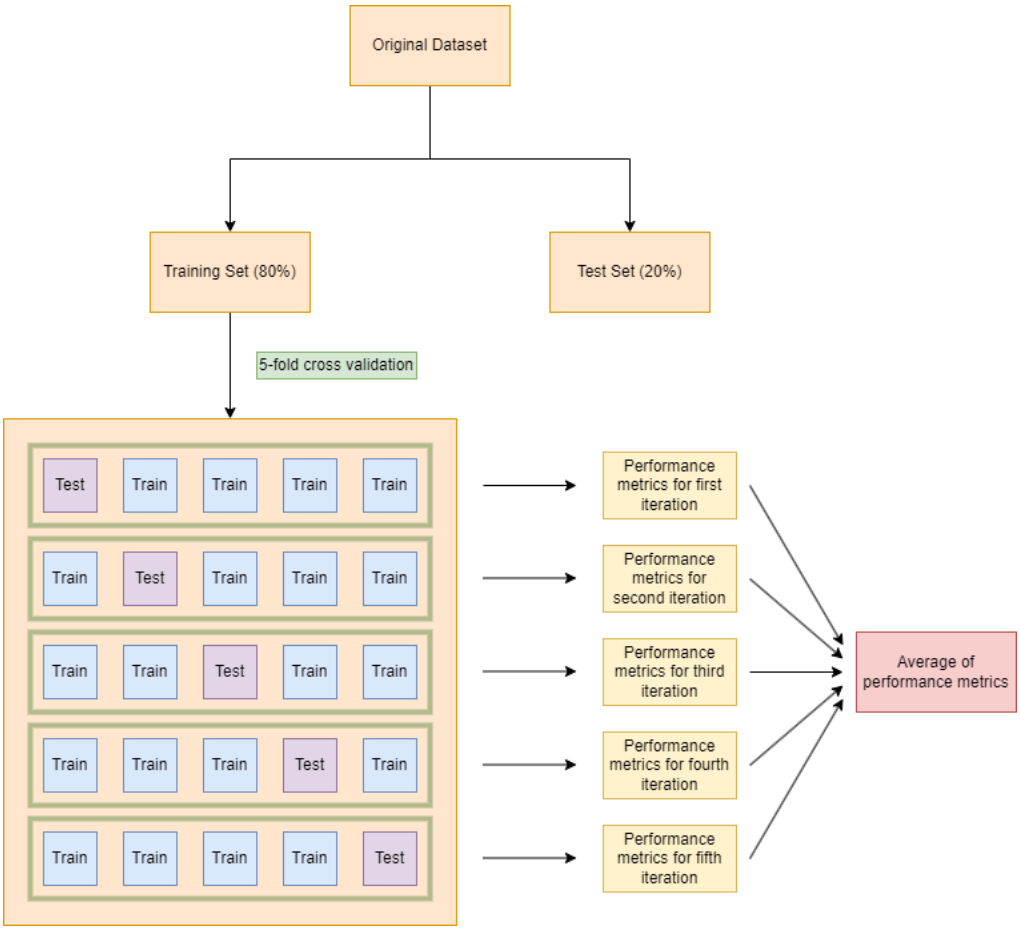

Figure 3.25: Illustration of 5-fold cross-validation

The five models that will be used in this project are initialized in an array with default hyperparameters.

```
models = {
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "AdaBoost": AdaBoostRegressor(random_state=42),
    "XGBoost": XGBRegressor(random_state=42),
    "Bagging Regressor": BaggingRegressor(random_state=42),
    "ANN": MLPRegressor(random_state=42)
}
```

Figure 3.26: Code snippet of initializing the five models in an array

Firstly, the models are trained without optimization techniques with 5-fold cross validation. The results produced from this training set will serve as a

baseline. The code snippet below shows the function defined to train a model without optimization techniques.

```
def evaluate_model_cv(model, x, y, cv=5):
    r2_scores = cross_val_score(model, x, y, cv=cv, scoring='r2')
    mse_scores = cross_val_score(model, x, y, cv=cv, scoring='neg_mean_squared_error')
    rmse_scores = np.sqrt(-mse_scores)
    return r2_scores.mean(), -mse_scores.mean(), rmse_scores.mean()

cv_results = {}

for name, model in models.items():
    r2_mean, mse_mean, rmse_mean = evaluate_model_cv(model, x_train, y_train)
    cv_results[name] = {
        "CV R2": r2_mean,
        "CV MSE": mse_mean,
        "CV RMSE": rmse_mean,
    }
    print(f"{name} CV R2: {cv_results[name]['CV R2']}")
    print(f"{name} CV MSE: {cv_results[name]['CV MSE']}")
    print(f"{name} CV RMSE: {cv_results[name]['CV RMSE']}")
```

Figure 3.27: Code snippet of training model with 5-fold cross validation

Before training the models with the best hyperparameters, GridSearchCV is used to find the optimal hyperparameters for each of the models. The hyperparameter grids for each of models are defined and shown in the figure below.

```
param_grids = {
    "Decision Tree": {'max_depth': [3, 5, 7, 10, 15],
                      'min_samples_leaf': [3, 5, 10, 15, 20],
                      'min_samples_split': [8, 10, 12, 18, 20, 16]},
    "AdaBoost": {'n_estimators': [50, 70, 90, 120, 180, 200],
                 'learning_rate': [0.001, 0.01, 0.1, 1, 10]},
    "XGBoost": {'n_estimators': [50, 100, 200],
                'max_depth': [3, 5, 7],
                'learning_rate': [0.1, 0.01, 0.001],
                'subsample': [0.5, 0.7, 1]},
    "Bagging Regressor": {'n_estimators': [300, 400, 500, 600, 700, 800],
                          'max_features': [0.90, 0.92, 0.95, 1.0],
                          'bootstrap': [True, False],
                          'bootstrap_features': [True, False]},
    "ANN": {'hidden_layer_sizes': [(i,) for i in range(2, 20)],
            'learning_rate': ['constant', 'invscaling', 'adaptive'],
            'alpha': [0.0001, 0.001, 0.01],
            'early_stopping': [True, False],
            'max_iter': [1000]}

}
```

Figure 3.28: Code snippet of defining hyperparameter grids

The hyperparameter grids defined are then used as a search space for the GridSearchCV to find the best hyperparameters for each of the models. The models with the best hyperparameters found are then stored inside an array for later use if needed.

```
best_models = {}

for name, model in models.items():
    grid_search = GridSearchCV(model, param_grids[name], cv=5, scoring='r2')
    grid_search.fit(x_train, y_train)
    best_models[name] = grid_search.best_estimator_
    print(f"Best parameters for {name}: {grid_search.best_params_}")
```

Figure 3.29: Code snippet of finding best hyperparameters with GridSearchCV

After that, the models are trained with the default hyperparameters and with the best hyperparameters found using GridSearchCV. The results for each of the models are compared.

```
test_results_before_tuning = {}
for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    test_results_before_tuning[name] = {"R2": r2, "MSE": mse, "RMSE": rmse}
    print(f"{name}: R2={r2:.4f}, MSE={mse:.4f}, RMSE={rmse:.4f}")

test_results_after_tuning = {}
for name, model in best_models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    test_results_after_tuning[name] = {"R2": r2, "MSE": mse, "RMSE": rmse}
    print(f"{name} (Tuned): R2={r2:.4f}, MSE={mse:.4f}, RMSE={rmse:.4f}")
```

Figure 3.30: Code snippet of training models with default hyperparameters and best hyperparameters found with GridSearchCV

The first feature selection technique to be experimented with is Principle Component Analysis (PCA). The training set of the data is fit to PCA

and a Cumulative Explained Variance Plot is plotted to decide on the number of principle components (features) to retain for capturing a sufficient amount of variance in the data.

```
pca = PCA().fit(x_train)

plt.figure(figsize=(10, 6))
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance by Number of Components')
plt.grid(True)
plt.show()
```

Figure 3.31: Code snippet for plotting Cumulative Explained Variance Plot

The Cumulative Explained Variance Plot shows that as the cumulative explained variance does not show significant increase after 5 number of components.

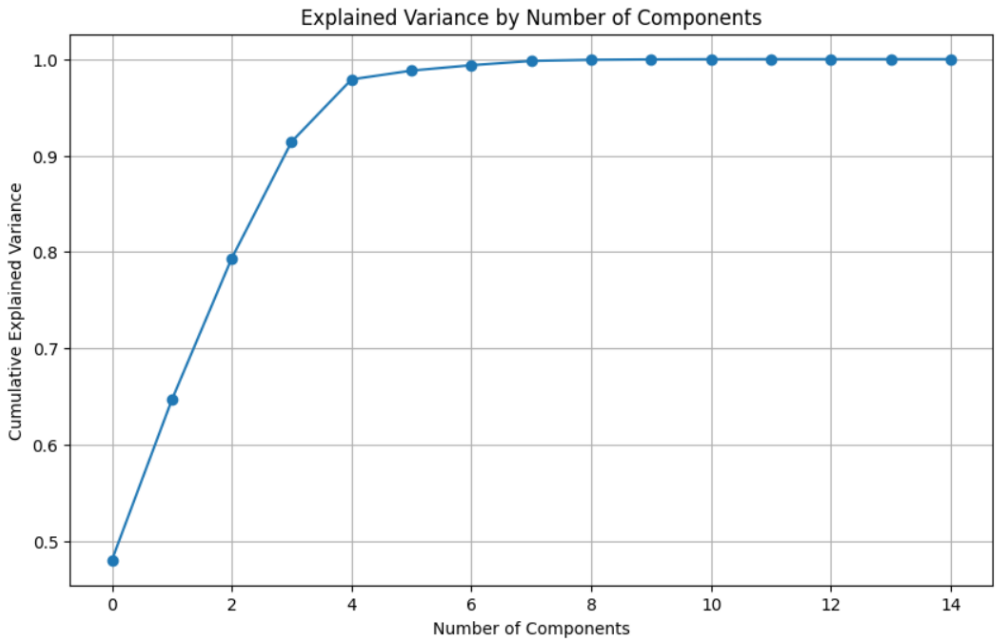

Figure 3.32: Cumulative Explained Variance Plot

The ratio of the cumulative explained variance for each number of components is also observed. A threshold of 95% is set to determine the optimal number of components to be retained in the data. Hence, the least number of

components that is able to achieve 95% variance explained is used. The results show that the optimal number of components to explain 95% variance of the data is 5 number of components.

```
Cumulative explained variance ratios:
1 components: 0.4803
2 components: 0.6465
3 components: 0.7935
4 components: 0.9144
5 components: 0.9790
6 components: 0.9881
7 components: 0.9937
8 components: 0.9982
9 components: 0.9993
10 components: 0.9998
11 components: 0.9999
12 components: 1.0000
13 components: 1.0000
14 components: 1.0000
15 components: 1.0000
Optimal number of components to explain 95.0% variance: 5
```

Figure 3.33: Cumulative explained variance ratios of each number of components

The training and test set are then reduced to 5 number of components and the best hyperparameters for each model on the new training and test set are found using GridSearchCV.

```
pca = PCA(n_components=5)
x_train_pca = pca.fit_transform(x_train)
x_test_pca = pca.transform(x_test)
best_models_pca = {}
for name, model in models.items():
    grid_search = GridSearchCV(estimator=model,
                               param_grid=param_grids[name],
                               scoring='r2',
                               cv=5,
                               n_jobs=-1)
    grid_search.fit(x_train_pca, y_train)
    best_models_pca[name] = grid_search.best_estimator_
    print(f"Best parameters for {name}: {grid_search.best_params_}")
```

Figure 3.34: Code snippet of finding best hyperparameters for each model with reduced training dataset using GridSearchCV

The models with the best hyperparameters are trained with the reduced training set and tested with the reduced test set.

```
test_results_pca = {}

for name, model in best_models_pca.items():
    model.fit(x_train_pca, y_train)
    y_pred_pca = model.predict(x_test_pca)
    r2_pca = r2_score(y_test, y_pred_pca)
    mse_pca = mean_squared_error(y_test, y_pred_pca)
    rmse_pca = np.sqrt(mse_pca)

    test_results_pca[name] = {
        "R2": r2_pca,
        "MSE": mse_pca,
        "RMSE": rmse_pca
    }
    print(f"{name} (With PCA): R2={r2_pca:.4f}, MSE={mse_pca:.4f}, RMSE={rmse_pca:.4f}")
```

Figure 3.35: Code snippet for training models on reduced training dataset with best hyperparameters

The next feature selection technique used is Boruta. The Random Forest model is used as an estimator for Boruta due to its ability to measure feature importance. The training and test sets are reduced based on the features that Boruta has selected.

```
forest = RandomForestRegressor(n_jobs=-1, max_depth=5)
boruta = BorutaPy(estimator=forest, n_estimators='auto', verbose=2, random_state=42)
boruta.fit(np.array(x_train), np.array(y_train))
if isinstance(x_train, pd.DataFrame):
    selected_features = x_train.columns[boruta.support_].to_list()
else:
    selected_features = [f'Feature {i}' for i in np.where(boruta.support_)[0]]
```

Figure 3.36: Code snippet for using Boruta to find the most important features

The best hyperparameters for the reduced dataset is found using GridSearchCV.

```
best_models_boruta = {}
for name, model in models.items():
    grid_search = GridSearchCV(estimator=model,
                               param_grid=param_grids[name],
                               scoring='r2',
                               cv=5,
                               n_jobs=-1)

    grid_search.fit(x_train_boruta, y_train)
    best_models_boruta[name] = grid_search.best_estimator_
```

Figure 3.37: Code snippet for finding best hyperparameters for dataset reduced with features selected from Boruta

The models are then trained using the best hyperparameters on the reduced training set.

```
test_results_boruta = {}

for name, model in best_models_boruta.items():
    model.fit(x_train_boruta, y_train)
    y_pred = model.predict(x_test_boruta)
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    test_results_boruta[name] = {
        "R2": r2,
        "MSE": mse,
        "RMSE": rmse
    }
    print(f"{name} (Boruta): R2={r2:.4f}, MSE={mse:.4f}, RMSE={rmse:.4f}")
```

Figure 3.38: Code snippet for training the models with best hyperparameters found from GridSearchCV on the reduced dataset with Boruta

The final feature selection technique that is included is LASSO. The training set is fit with LASSO and it is then used to reduce the training and test set.

```
lasso = Lasso(alpha=0.01)
lasso.fit(x_train, y_train)
selected_features_lasso = x.columns[(lasso.coef_ != 0)].to_list()
x_train_lasso = x_train[:, lasso.coef_ != 0]
x_test_lasso = x_test[:, lasso.coef_ != 0]
```

Figure 3.39: Code snippet for reducing training and test set with LASSO selected features

The models with the best hyperparameters using the LASSO reduced dataset are found using GridSearchCV.

```
for name, model in models.items():
    grid_search = GridSearchCV(estimator=model,
                               param_grid=param_grids[name],
                               scoring='r2',
                               cv=5,
                               n_jobs=-1)

    grid_search.fit(x_train_lasso, y_train)
    best_models_lasso[name] = grid_search.best_estimator_
```

Figure 3.40: Code snippet for finding best hyperparameters for dataset reduced with LASSO

The models are then trained with the best hyperparameters found with GridSearchCV on the LASSO reduced training set.

```
for name, model in best_models_lasso.items():
    model.fit(x_train_lasso, y_train)
    y_pred = model.predict(x_test_lasso)
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    test_results_lasso[name] = {
        "R2": r2,
        "MSE": mse,
        "RMSE": rmse
    }
    print(f"{name} (Lasso): R2={r2:.4f}, MSE={mse:.4f}, RMSE={rmse:.4f}")
```

Figure 3.41: Code snippet for training the models with best hyperparameters found from GridSearchCV on the LASSO reduced dataset

3.3.5 Model Testing

After using 5-fold cross-validation on the models, the dataset will automatically be partitioned into 5 equal-sized folds. The models are then trained and evaluated 5 times. In each iteration, the models are trained on 4 folds and the remaining fold is used to evaluate the performance of the models. Hence, the models will be tested 5 times during 5-fold cross-validation to gain rough estimates of the models' performances in a real world context. The models with default hyperparameters, best hyperparameters and different feature selection techinques are all tested on their respective test sets to verify the models' performances.

3.3.6 Model Performance Evaluation

The performance of each of the models with the different optimization techniques used are evaluated based on the R^2 score, RMSE and MSE. For 5-fold cross validation, the performance metrics are generated by calculating the mean of each metric through the five splits of cross-validation.

```
for name, model in models.items():
    r2_mean, mse_mean, rmse_mean = evaluate_model_cv(model, x_train, y_train)
    cv_results[name] = {
        "CV R2": r2_mean,
        "CV MSE": mse_mean,
        "CV RMSE": rmse_mean,
    }
    print(f"{name} CV R2: {cv_results[name]['CV R2']}")
    print(f"{name} CV MSE: {cv_results[name]['CV MSE']}")
    print(f"{name} CV RMSE: {cv_results[name]['CV RMSE']}")
```

Figure 3.42: Code snippet for evaluating model performance with cross-validation

For each of the different optimization techniques used, the models are tested on the test set.

```
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
test_results_lasso[name] = {
    "R2": r2,
    "MSE": mse,
    "RMSE": rmse
}
print(f"{name} (Lasso): R2={r2:.4f}, MSE={mse:.4f}, RMSE={rmse:.4f}")
```

Figure 3.43: Code snippet for evaluating models' performances on test set

3.3.7 Selection of Best Performing Model

The best performing model is selected based on the highest performance metrics. As the performance metrics may vary from different models with different optimization techniques, the model that performs well across different optimization techniques is selected.

3.3.8 Generate Objective-functions from the Best Performing Model

Using the best model selected, three different objective-functions are generated to be used in two experiments as requested by Dr. Lim based on varying values of the 'ES/C' column which stands for eggshell percentage per concrete:

(i) A single objective-function for concretes with and without eggshell percentage (0%-10%)

(ii) Set of two objective-functions: (i) objective-function for concretes with eggshell (more than 0%) and (ii) objective-function for concretes without eggshell (0%)

The reason for generating three different objective-functions based on the eggshell percentage of concrete is because the dataset used in this project consists of samples of lightweight concretes with and without eggshell percentages. Hence, the two experiments are carried out to find out the best set of objective-functions. The approved set of objective-functions will be used in a PSO algorithm to find the best combinations of feature values to maximize the strength of concrete. As most of the data mining models used in this project are complex and do not directly provide usable equations, a hybrid approach that fits a linear regression model to the predictions of a model on the training data to create an approximation of the model's output is used.

```
predictions = br.predict(x_train)
lin_model = LinearRegression()
lin_model.fit(x_train, predictions)

print("Linear Regression Coefficients:", lin_model.coef_)
print("Intercept:", lin_model.intercept_)
```

Figure 3.44: Code snippet of the hybrid approach

3.3.9 Obtain verification of Objective-function

The objective-functions generated are passed to Dr. Lim Siong Kang, a civil engineering lecturer at UTAR for verification. The objective-functions are verified with a sample dataset prepared by him which consists of 24 sample data. Each of the set of objective-functions are used to generate values for the strength of concrete of each sample data and the generated values are compared with the actual values. The set of objective-functions are evaluated based on the Mean Absolute Error (MAE), maximum absolute difference between actual and

predicted value, minimum absolute difference between actual and predicted value, median of absolute difference between actual and predicted value and standard deviation of absolute difference between actual and predicted value. Based on Dr. Lim's feedback, one of the set of objective-functions will be selected to be used for the PSO algorithm.

3.3.10 Implementation of Particle Swarm Optimization to solve Single Objective Optimization Problem

The selected objective-function is used in this step to solve a single objective optimization problem which is to maximize the strength of concrete. To ensure that the optimal values of each feature found by the PSO algorithm are acceptable in the construction industry, constraints and boundaries are set for each of the feature values. The objective-function to be used in the algorithm is first defined. The PSO algorithm is designed to minimize the objective-function by default. Hence, it is necessary to negate strength to maximize the objective-function.

```python
def objective_function(x):
    w_c = x[:, 0]
    ES_C = x[:, 1]
    IvST_dia = np.round(x[:, 2])
    pb_mortar_density = np.round(x[:, 3])
    V_foam_theo = x[:, 4]
    mass_foam_added = x[:, 5]
    target_r = np.round(x[:, 6])
    fresh_r = np.clip(np.round(x[:, 7]), target_r - 50, target_r + 50)
    HD = np.clip(np.round(x[:, 8]), target_r - 50, target_r + 50)
    SSD_D = np.clip(np.round(x[:, 9]), target_r - 200, target_r + 200)
    OD_D = np.clip(np.round(x[:, 10]), target_r - 200, target_r + 200)
    type_mould_3G = np.round(x[:, 11]).astype(int)
    type_mould_sm = np.round(x[:, 12]).astype(int)
    age_7 = np.round(x[:, 13]).astype(int)
    age_28 = np.round(x[:, 14]).astype(int)
    type_mould_3G = np.clip(type_mould_3G, 0, 1)
    type_mould_sm = np.clip(type_mould_sm, 0, 1)
    age_7 = np.clip(age_7, 0, 1)
    age_28 = np.clip(age_28, 0, 1)

    strength = (
        3.320178633391417
        + (0.44114704 * w_c)
        - (0.08720103 * ES_C)
        + (0.02819477 * IvST_dia)
        - (0.48016159 * pb_mortar_density)
        + (5.18345732 * V_foam_theo)
        + (1.09592118 * mass_foam_added)
        - (1.68066183 * target_r)
        + (8.23721416 * fresh_r)
        + (1.23358836 * HD)
        - (0.75241553 * SSD_D)
        + (1.54295495 * OD_D)
        + (0.06370227 * type_mould_3G)
        - (0.06370227 * type_mould_sm)
        - (0.22541453 * age_7)
        + (0.22541453 * age_28)
    )
    return -strength
```

Figure 3.45: Code snippet of defining objective-function

```
bounds = [
    (0.5, 0.8),
    (0.00, 0.10),
    (500, 800),
    (1900, 2100),
    (20, 60),
    (1, 5),
    (800, 1400),
    (750, 1450),
    (750, 1450),
    (600, 1600),
    (600, 1600),
    (0, 1),
    (0, 1),
    (0, 1),
    (0, 1),
]

lower_bounds = np.array([bound[0] for bound in bounds])
upper_bounds = np.array([bound[1] for bound in bounds])

bounds = (lower_bounds, upper_bounds)
```

Figure 3.46: Bounds for each feature

Using the defined objective-function, the best hyperparameters for the PSO algorithm are found.

```
param_grid = {
    'n_particles': [20, 50, 100],
    'w': [0.5, 0.7, 1.0],
    'c1': [1.5, 2.0, 2.5],
    'c2': [1.5, 2.0, 2.5],
    'max_iter': [100, 200, 300]
}
best_combination = None
best_cost = float('inf')
for n_particles, w, c1, c2, max_iter in product(
    param_grid['n_particles'],
    param_grid['w'],
    param_grid['c1'],
    param_grid['c2'],
    param_grid['max_iter']
):

    options = {'c1': c1, 'c2': c2, 'w': w}
    optimizer = ps.single.GlobalBestPSO(
        n_particles=n_particles,
        dimensions=15,
        options=options,
        bounds=bounds
    )
    cost, pos = optimizer.optimize(objective_function, iters=max_iter, verbose=False)
    if cost < best_cost:
        best_cost = cost
        best_combination = (n_particles, w, c1, c2, max_iter)
print("Best hyperparameter combination:", best_combination)
```

Figure 3.47: Code snippet for finding best hyperparameters for PSO

The best hyperparameters found and the objective-function are then used in the PSO algorithm to generate the optimal values of each feature that maximizes the strength of concrete.

```
n_particles, w, c1, c2, max_iter = best_combination

optimizer = ps.single.GlobalBestPSO(
    n_particles=n_particles,
    dimensions=15,
    options={'c1': c1, 'c2': c2, 'w': w},
    bounds=bounds
)

best_cost, best_pos = optimizer.optimize(objective_function, iters=max_iter)

print("Best position (optimal feature values):", best_pos)
```

Figure 3.48: Code snippet for PSO algorithm

3.4 Evaluation Metrics

The R^2 score is used as the performance metric to evaluate the performance of the models with different optimization techniques applied. Besides R^2 score, MSE and RSME will also be evaluated to gain a deeper insight of the performance of the models. The R^2 score is prioritised over the other metrics such as MSE and RMSE when evaluating the models in this project. This is because a high R^2 score indicates that the model fits the data well and can capture the underlying patterns in the data. A model with high R^2 score is generally more capable in making accurate predictions. Low MSE and RMSE shows that the model's predictions are closer to that of the actual values. The performance of the models with and without optimization techniques will be evaluated to observe if the models perform better with optimization techniques. The results observed from each of the models with different optimization techniques are described in Section 4.2.

3.5 Python and Libraries

The implementation of the data mining models are conducted with the use of the Python language. The library resources of python used in this project is shown in the table below.

Table 3.1: Python Lybraries used

Library	Usage
Pandas	Handle data manipulation processes such as cleaning
NumPy	Scientific computation of numerical tasks with data
Seaborn	Create informative statistical graphs
Scipy	Additional functionality for scientific computations
Scikit-learn	Tools for building data mining models
Joblib	Save and load models

Boruta	Feature selection library to identify important features in a dataset
Pyswarms	Library for Particle Swarm Optimization algorithm
Itertools	Create efficient iterators for looping

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this section, the results obtained from each category of experiments for each of the models trained are presented. This project aims to evaluate different models for predicting concrete strength. The chosen models for this project are Decision Tree, AdaBoost, XGBoost, Bagging Regressor and Artificial Neural Network.

4.2 Results from 5-fold cross-validation

Table 4.1: Results of each model with 5-fold cross validation (CV) on training dataset

Model	Mean R^2 (CV)	Mean MSE (CV)	Mean RMSE (CV)
Decision Tree	0.9395	0.2963	0.5370
AdaBoost	0.9506	0.2411	0.4901
XGBoost	0.9498	0.2473	0.4951
Bagging Regressor	0.9585	0.2022	0.4477
Artificial Neural Network	0.9460	0.2610	0.5044

Table 4.1 shows the performance metrics of each model that are obtained through 5-fold cross validation on the training set which contains 80% of the data. It is observed that all the models are able to perform well and are not prone to overfitting. Hence, all the models will be trained and tested with hyperparameter tuning and feature selection techniques. The Bagging Regressor model is able to outperform all the other models as it recorded the highest mean R^2 and the lowest mean MSE and RMSE out of all the models with values of 0.9585, 0.2022 and 0.4477 respectively.

4.3 Results with Default Hyperparameters and Tuned Hyperparameters

Table 4.2: Results of Each Model with Default Hyperparameters and Tuned Hyperparameters Found With GridSearchCV on test set

Model	With default hyperparameters				With tuned hyperparameters found with GridSearchCV			
	R^2	MSE	RMSE	Default Hyperparameters	R^2	MSE	RMSE	Tuned Hyperparameters
DT	0.9208	0.3646	0.6038	{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}	0.9445	0.2556	0.5056	{'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 8}
AB	0.9454	0.2514	0.5014	{'learning_rate': 1, 'n_estimators': 50}	0.9453	0.2519	0.5019	{'learning_rate': 1, 'n_estimators': 90}
XGB	0.9467	0.2456	0.4956	{'learning_rate': 0.3, 'max_depth': 6, 'n_estimators': 100, 'subsample': 1}	0.9442	0.2569	0.5069	{'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 50, 'subsample': 0.7}

BR	**0.9471**	**0.2436**	**0.4935**	{'bootstrap': True, 'bootstrap_features': False, 'max_features': 1.0, 'n_estimators': 10}	**0.9516**	**0.2227**	**0.4719**	{'bootstrap': True, 'bootstrap_features': True, 'max_features': 0.9, 'n_estimators': 300}
ANN	0.9434	0.2607	0.5106	{'alpha': 0.0001, 'early_ stopping': False, 'hidden_ layer_sizes': (100,), 'learning_ rate': 'constant', 'max_iter': 200}	0.9392	0.2801	0.5292	{'alpha': 0.01, 'early_stopping': False, 'hidden_layer_sizes': (17,), 'learning_rate': 'constant', 'max_iter': 1000}

DT – Decision Tree

AB – AdaBoost

XGB – XGBoost

BR – Bagging Regressor

*ANN – Artificial Neural Network

Table 4.2 shows the performance metrics obtained for each model with default hyperparameters and with tuned hyperparameters found by using GridSearchCV. It is observed that the Decision Tree and Bagging Regressor models perform better with the tuned hyperparameters found by using GridSearchCV. AdaBoost, XGBoost and Artificial Neural Network have shown to perform worse with tuned hyperparameters. This could be due to the tuned hyperparameters leading to overly complex models which are unable to generalize well. The BR model performs the best out of all the models with and without tuned hyperparameters. It has recorded a R^2 score of 0.9471, MSE of 0.2436 and RMSE of 0.4935 with default hyperparameters. With tuned hyperparameters, it is able to record a better R^2 score of 0.9516, MSE of 0.2227 and RMSE of 0.4719. Hence, it is concluded that the BR model is the best performing model when trained and tested on the full dataset.

4.4 Results with Different Feature Selection Techniques

Table 4.3: Results of Each Model With Different Feature Selection Techniques on test set

Model	With PCA (n_components =5)			With Boruta			With LASSO		
	R^2	MSE	RMSE	R^2	MSE	RMSE	R^2	MSE	RMSE
DT	**0.9497**	**0.2314**	**0.4810**	0.9500	0.2302	0.4798	0.9441	0.2575	0.5074
	{'max_depth': 7, 'min_samples_leaf': 5, 'min_samples_split': 20}								
AB	0.9298	0.3233	0.5686	0.9433	0.2612	0.5111	0.9428	0.2633	0.5132
XGB	0.9440	0.2579	0.5078	0.9466	0.2461	0.4960	0.9438	0.2520	0.5111
BR	0.9416	0.2690	0.5186	**0.9504**	**0.2284**	**0.4779**	**0.9525**	**0.2187**	**0.4677**
				{'bootstrap: True, 'bootstrap_features': True, 'max_features': 1.0, 'n_estimators': 300}			{'bootstrap: True, 'bootstrap_features': True, 'max_features': 1.0, 'n_estimators': 300}		
ANN	0.9457	0.2502	0.5002	0.9316	0.3146	0.5610	0.9477	0.2407	0.4906

DT – Decision Tree

AB – AdaBoost

XGB – XGBoost

BR – Bagging Regressor

ANN – Artificial Neural Network

Table 4.3 shows the metrics of each model recorded with three different types of feature selection techniques (PCA, Boruta and LASSO). Generally, the models do not perform significantly better when trained with feature selection techniques. The Decision Tree model is able to perform the best out of all the models with PCA. It has recorded the best R^2 score, MSE and RMSE with values of 0.9497, 0.2314 and 0.4810 respectively. However, with Boruta and LASSO, the Bagging Regressor model outperforms all the other models. With Boruta, it has recorded a R^2 score, MSE and RMSE with values of 0.9504, 0.2284 and 0.4779 respectively. Compared to Boruta, the BR model performs better with LASSO as it has recorded better performance metrics with a R^2 score of 0.9525, MSE of 0.2187 and RMSE of 0.4677. The results show that there is a slight performance loss when the BR model is trained with Boruta. However, there is a slight performance gain when it is trained with LASSO. Hence, it is concluded that the BR model with LASSO is the best performing model and is suitable to be used for concrete strength prediction.

4.5 Results from Different Set of Objective-functions

Generally, the BR model has shown to outperform the other models with default hyperparameters, tuned hyperparameters and different feature selection techniques. The BR model performs the best with LASSO but it is not chosen to generate objective-functions as the full set of features is needed in order to generate the optimal values for each feature in the PSO algorithm. Hence, the BR model with tuned hyperparameters is used in the PSO algorithm.

4.5.1 Single Objective-function

Equation (4.1) shows the objective-function defined for concretes with any percentages of eggshell.

$$\begin{aligned} Strength = {} & 3.32 + (0.44 * wc) - (0.09 * esc) + (0.03 * ivstdia) \\ & - (0.48 * pmd) + (5.18 * vft) + (1.10 * mfa) - (1.68 * tr) \\ & + (8.24 * fr) + (1.23 * hd) + (0.75 * ssd) + (1.54 * odd) \\ & + (0.06 * 3g) - (0.06 * sm) - (0.23 * age7) + (0.23 \\ & * age28) \end{aligned} \tag{4.1}$$

where

wc = water to cement ratio

esc = eggshell percentage per concrete

$ivstdia$= IvST. diameter

pmd = ρb mortar density

vft = V foam (theo) % of total volume

mfa = Mass of foam added (% of total mass)

tr = target density

fr = fresh density

hd = hardened density

ssd = saturated surface dry density

odd = oven dry density

$3g$ = 3G type of mould used

sm = sm type of mould used

$age7$ = 7 days of curing concrete

$age28$ = 28 days of curing concrete

Table 4.4: Results of Objective-function for Concretes With Eggshell

Actual value of strength	Predicted value of strength	Absolute difference between actual and predicted value of strength
5.15	4.95	0.2
4.50	4.71	0.31
5.06	4.72	0.34
5.33	3.86	1.47
4.89	4.08	0.81
5.01	3.82	1.19
2.38	2.17	0.21
2.85	2.54	0.31
2.37	2.28	0.09
1.92	1.28	0.64
1.94	1.26	0.68
2.22	1.31	0.91
6.87	5.77	1.10
6.39	5.68	0.71
6.37	5.67	0.70
6.70	4.69	2.01
7.42	4.87	2.55
6.99	4.81	2.18
3.19	3.24	0.05
2.45	2.88	0.43
2.79	3.18	0.39
3.10	2.52	0.58
2.28	2.41	0.13

2.77	2.13	0.64

Table 4.5: Performance Metrics of Objective-function for Concretes With Eggshell

MAE of difference between actual and predicted value	0.78
Minimum of difference between actual and predicted value	0.05
Maximum of difference between actual and predicted value	2.55
Median of difference between actual and predicted value	0.66
Standard Deviation of difference between actual and predicted value	0.66

The results show that the objective-function is able to perform well with a MAE of 0.78. It has recorded a minimum of absolute difference between actual and predicted value of 0.05. However, the maximum of absolute difference between actual and predicted value recorded is 2.55. This may be due to outliers that are present in the sample dataset used for verification. The objective-function has recorded a median and standard deviation of absolute difference between actual and predicted value of 0.66.

4.5.2 Set of Two Objective-functions

Equation (4.2) shows the objective-function defined for concretes without eggshell (0%).

$$\begin{aligned} Strength = \ & 3.07 - (0.76 * wc) + (0.49 * ivstdia) - (3.00 * pmd) \\ & + (25.42 * vft) + (1.03 * mfa) - (1.52 * tr) + (29.18 * fr) \\ & + (0.43 * hd) - (0.78 * ssd) + (2.33 * odd) + (0.08 * 3g) \\ & - (0.08 * sm) - (0.24 * age7) + (0.24 * age28) \end{aligned} \quad (4.2)$$

where

wc = water to cement ratio

$ivstdia$= IvST. diameter

pmd = ρb mortar density

vft = V foam (theo) % of total volume

mfa = Mass of foam added

tr = target density

fr = fresh density

hd = hardened density

ssd = saturated surface dry density

odd = oven dry density

$3g$ = 3G type of mould used

sm = sm type of mould used

$age7$ = 7 days of curing concrete

$age28$ = 28 days of curing concrete

Equation (4.3) shows the objective-function defined for concretes with eggshell (more than 0%).

$$\begin{aligned} Strength = {} & 3.42 + (0.08 * wc) - (0.08 * esc) + (0.20 * ivstdia) \\ & - (1.19 * pmd) + (10.15 * vft) + (0.97 * mfa) \\ & - (0.02 * tr) + (11.10 * fr) + (1.13 * hd) - (0.78 * ssd) \\ & + (2.01 * odd) + (0.06 * 3g) - (0.06 * sm) - (0.22 * age7) \\ & + (0.22 * age28) \end{aligned} \tag{4.3}$$

where

wc = water to cement ratio

esc = eggshell percentage per concrete

$ivstdia$= IvST. diameter

pmd = ρb mortar density

vft = V foam (theo) % of total volume

mfa = Mass of foam added

tr = target density

fr = fresh density

hd = hardened density

ssd = saturated surface dry density

odd = oven dry density

$3g$ = 3G type of mould used

sm = sm type of mould used

$age7$ = 7 days of curing concrete

$age28$ = 28 days of curing concrete

Table 4.6: Results of Set of Two Objective-functions: (i) Objective-function for Concretes Without Eggshell (0%) and (ii) Objective-function for Concretes With Eggshell (more than 0%)

Actual value of strength	Predicted value of strength	Absolute difference between actual and predicted value of strength
5.15	5.25	0.10
4.50	5.09	0.59
5.06	5.03	0.03
5.33	3.95	1.38
4.89	4.18	0.71
5.01	3.90	1.11
2.38	2.74	0.36
2.85	3.08	0.23
2.37	2.86	0.49
1.92	1.48	0.44

1.94	1.46	0.48
2.22	1.48	0.74
6.87	6.01	0.86
6.39	5.98	0.41
6.37	6.00	0.37
6.70	4.74	1.96
7.42	4.92	2.50
6.99	4.87	2.12
3.19	3.93	0.74
2.45	3.57	1.12
2.79	3.82	1.03
3.10	2.42	0.68
2.28	2.34	0.06
2.77	2.21	0.56

Table 4.7: Performance Metrics of Set of Two Objective-functions

MAE of absolute difference between actual and predicted value	0.80
Minimum of absolute difference between actual and predicted value	0.03
Maximum of absolute difference between actual and predicted value	2.50

Median of absolute difference between actual and predicted value	0.62
Standard Deviation of absolute difference between actual and predicted value	0.63

Compared to the single objective function used in Section 4.5.1, the set of two objective-functions perform slightly worse with a MAE of 0.80. However, it has recorded a better minimum and maximum of absolute difference between actual and predicted value with values of 0.03 and 2.50 respectively. The median and standard deviation of absolute difference between actual and predicted value recorded are 0.62 and 0.63 respectively.

4.6 Results from Particle Swarm Optimization

Table 4.8: Optimal Values for Each Feature Generated from PSO with tuned hyperparameters

Water to cement ratio	0.73
Eggshell percentage per concrete	0.02
IvST. Diameter	680
Density of mortar portion	1960
Percentage of foam	56.8
Mass of foam added (% of total mass)	1.72
Target density	1400
Fresh density	1427
Hardened density	1287
Saturated surface dry density	1200
Oven dry density	1555
3G type of mould	1
sm type of mould	0
7 days of curing concrete	0
28 days of curing concrete	1
Maximum strength achievable	27.96

Best hyperparameters	{‘n_particles’: 100, ‘w’: 1.0, ‘c1’: 1.5, ‘c2’: 1.5, ‘max_iter’: 300}

Based on Table 4.8, the maximum strength that is achievable with the optimal values of each feature generated by the PSO algorithm is 27.96. The best hyperparameters used for the PSO algorithm is {‘n_particles’: 100, ‘w’: 1.0, ‘c1’: 1.5, ‘c2’: 1.5, ‘max_iter’: 300}.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

By achieving the objectives defined, this project is able to provide the construction industry with a more efficient and accurate method to predict strength of concrete. Unlike conventional methods that rely heavily on empirical calculations and extensive laboratory testing, data mining techniques do not require manpower.

The results show that the Bagging Regressor performs the best overall with hyperparameter tuning and different feature selection techniques. It is able to record a R^2 score of 0.9525 with LASSO and tuned hyperparameters. Hence, it is selected to be used in the PSO algorithm.

The PSO algorithm is able to successfully generate a combination of optimal feature values that maximizes the strength of concrete. The maximum strength of concrete achievable found by PSO is 27.96. This result demonstrates the effectiveness of the PSO algorithm in optimizing the concrete mixtures.

5.2 Recommendations

Future enhancements can be made to the project by requesting for a larger dataset to be prepared beforehand as the process of curing concretes to get the results of strength of concretes may take up to several months. This can help to overcome the limitations of this project as there was insufficient training data provided which may have caused some inaccuracy in the results.

Another suggestion would be to apply oversampling techniques that are suitable for regression tasks such as Adaptive Synthetic Sampling for Regression (ADASYN-R). However, oversampling techniques such as

ADASYN-R are designed for imbalanced learning in regression tasks which is not present in the dataset used.

REFERENCES

Ahmad, A., Ahmad, W., Aslam, F. and Joyklad, P., 2022. Compressive strength prediction of fly ash-based geopolymer concrete via advanced machine learning techniques. *Case Studies in Construction Materials*, *16*, p.e00840.

Ahmad, A., Farooq, F., Niewiadomski, P., Ostrowski, K., Akbar, A., Aslam, F. and Alyousef, R., 2021. Prediction of compressive strength of fly ash based concrete using individual and ensemble algorithm. *Materials*, *14*(4), p.794.

Ahmad, W., Ahmad, A., Ostrowski, K.A., Aslam, F., Joyklad, P. and Zajdel, P., 2021. Application of advanced machine learning approaches to predict the compressive strength of concrete containing supplementary cementitious materials. *Materials*, *14*(19), p.5762.

Aïtcin, P.C., 2000. Cements of yesterday and today: concrete of tomorrow. *Cement and Concrete research*, *30*(9), pp.1349-1359.

Asteris, P.G., Skentou, A.D., Bardhan, A., Samui, P. and Pilakoutas, K., 2021. Predicting concrete compressive strength using hybrid ensembling of surrogate machine learning models. *Cement and Concrete Research*, *145*, p.106449.

Behnood, A. and Golafshani, E.M., 2020. Machine learning study of the mechanical properties of concretes containing waste foundry sand. *Construction and Building Materials*, *243*, p.118152.

Biswal, A. (2021). *What is Bagging in Machine Learning And How to Perform Bagging*. [online] Simplilearn. Available at: https://www.simplilearn.com/tutorials/machine-learning-tutorial/bagging-in-machine-learning [Accessed 1 Apr. 2024].

Carmona, P., Climent, F. and Momparler, A., 2019. Predicting failure in the US banking sector: An extreme gradient boosting approach. *International Review of Economics & Finance*, *61*, pp.304-323.

Chaabene, W.B., Flah, M. and Nehdi, M.L., 2020. Machine learning prediction of mechanical properties of concrete: Critical review. *Construction and Building Materials*, *260*, p.119889.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I. and Zhou, T., 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4), pp.1-4.

Chicco, D., Warrens, M.J. and Jurman, G., 2021. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *Peerj computer science*, *7*, p.e623.

Dhaliwal, S.S., Nahid, A.A. and Abbas, R., 2018. Effective intrusion detection system using XGBoost. *Information*, *9*(7), p.149.

Dongare, A.D., Kharde, R.R. and Kachare, A.D., 2012. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, *2*(1), pp.189-194.

Farooq, F., Nasir Amin, M., Khan, K., Rehan Sadiq, M., Javed, M.F., Aslam, F. and Alyousef, R., 2020. A comparative study of random forest and genetic

engineering programming for the prediction of compressive strength of high strength concrete (HSC). *Applied Sciences*, *10*(20), p.7330.

Feng, D.C., Liu, Z.T., Wang, X.D., Chen, Y., Chang, J.Q., Wei, D.F. and Jiang, Z.M., 2020. Machine learning-based compressive strength prediction for concrete: An adaptive boosting approach. *Construction and Building Materials*, *230*, p.117000. 950.

Figueiredo Filho, D.B., Júnior, J.A.S. and Rocha, E.C., 2011. What is R2 all about?. *Leviathan (São Paulo)*, (3), pp.60-68.Haque, M.N., Al-Khaiat, H. and Kayali, O., 2004. Strength and durability of lightweight concrete. *Cement and Concrete Composites*, *26*(4), pp.307-314.

Hodson, T.O., Over, T.M. and Foks, S.S., 2021. Mean squared error, deconstructed. *Journal of Advances in Modeling Earth Systems*, *13*(12), p.e2021MS002681.

Jaadi, Z. (2019). *When and Why to Standardize Your Data?* [online] Built In. Available at: https://builtin.com/data-science/when-and-why-standardize-your-data [Accessed 7 Apr. 2024].

Jierula, A., Wang, S., OH, T.-M. and Wang, P. (2021). Study on Accuracy Metrics for Evaluating the Predictions of Damage Locations in Deep Piles Using Artificial Neural Networks with Acoustic Emission Data. *Applied Sciences*, 11(5), p.2314. doi:https://doi.org/10.3390/app11052314.

Juneja, M. and Nagar, S.K., 2016, October. Particle swarm optimization algorithm and its parameters: A review. In *2016 International Conference on Control, Computing, Communication and Materials (ICCCCM)* (pp. 1-5). IEEE.

Kaloop, M.R., Kumar, D., Samui, P., Hu, J.W. and Kim, D., 2020. Compressive strength prediction of high-performance concrete using gradient tree boosting machine. *Construction and Building Materials*, *264*, p.120198.

Kang, M.C., Yoo, D.Y. and Gupta, R., 2021. Machine learning-based prediction for compressive and flexural strengths of steel fiber-reinforced concrete. *Construction and Building Materials*, *266*, p.121117.

Lee, S.C., 2003. Prediction of concrete strength using artificial neural networks. *Engineering structures*, *25*(7), pp.849-857.

Ly, H.B., Nguyen, T.A. and Tran, V.Q., 2021. Development of deep neural network model to predict the compressive strength of rubber concrete. *Construction and Building Materials, 301*, p.124081.

Marini, F. and Walczak, B., 2015. Particle swarm optimization (PSO). A tutorial. *Chemometrics and Intelligent Laboratory Systems, 149*, pp.153-165.

Mohammed, A., Rafiq, S., Sihag, P., Kurda, R. and Mahmood, W., 2021. Soft computing techniques: systematic multiscale models to predict the compressive strength of HVFA concrete based on mix proportions and curing times. *Journal of Building Engineering, 33*, p.101851.

Moradi, M.J., Khaleghi, M., Salimi, J., Farhangi, V. and Ramezanianpour, A.M., 2021. Predicting the compressive strength of concrete containing metakaolin with different properties using ANN. *Measurement, 183*, p.109790.

Nguyen, H., Vu, T., Vo, T.P. and Thai, H.T., 2021. Efficient machine learning models for prediction of concrete strengths. *Construction and Building Materials, 266*, p.120

Nguyen, K.T., Nguyen, Q.D., Le, T.A., Shin, J. and Lee, K., 2020. Analyzing the compressive strength of green fly ash based geopolymer concrete using experiment and machine learning approaches. *Construction and Building Materials, 247*, p.118581.

Öztaş, A., Pala, M., Özbay, E., Kanca, E., Çagˇlar, N. and Bhatti, M.A., 2006. Predicting the compressive strength and slump of high strength concrete using neural network. *Construction and building materials*, *20*(9), pp.769-775.

Pedersen, M.E.H. and Chipperfield, A.J., 2010. Simplifying particle swarm optimization. *Applied Soft Computing*, *10*(2), pp.618-628.

Plevris, V., Solorzano, G., Bakas, N.P. and Ben Seghier, M.E.A., 2022, November. Investigation of performance metrics in regression analysis and machine learning-based prediction models. In *8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2022)*. European Community on Computational Methods in Applied Sciences.

Rainio, O., Teuho, J. and Klén, R., 2024. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1), p.6086.

Shahmansouri, A.A., Bengar, H.A. and Ghanbari, S., 2020. Compressive strength prediction of eco-efficient GGBS-based geopolymer concrete using GEP method. *Journal of Building Engineering, 31*, p.101326.

Shi, Y. and Eberhart, R.C., 1999, July. Empirical study of particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (Vol. 3, pp. 1945-1950). IEEE.

Song, H., Ahmad, A., Farooq, F., Ostrowski, K.A., Maślak, M., Czarnecki, S. and Aslam, F., 2021. Predicting the compressive strength of concrete with fly ash admixture using machine learning algorithms. *Construction and Building Materials*, *308*, p.125021.

Song, M.P. and Gu, G.C., 2004, August. Research on particle swarm optimization: a review. In *Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826)* (Vol. 4, pp. 2236-2241). IEEE.

Taherkhani, A., Cosma, G. and McGinnity, T.M., 2020. AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, *404*, pp.351-366.

Thienel, K.C., Haller, T. and Beuntner, N., 2020. Lightweight concrete—From basics to innovations. *Materials*, *13*(5), p.1120.

Tran, V.Q., Dang, V.Q. and Ho, L.S., 2022. Evaluating compressive strength of concrete made with recycled concrete aggregates using machine learning approach. *Construction and Building Nunez, I., Marani, A. and Nehdi, M.L., 2020. Mixture optimization of recycled aggregate concrete using hybrid machine learning model. Materials, 13(19), p.4331.Materials*, *323*, p.126578.

Wang, D., Tan, D. and Liu, L., 2018. Particle swarm optimization algorithm: an overview. *Soft computing*, *22*(2), pp.387-408

Zou, J., Han, Y. and So, S.S., 2009. Overview of artificial neural networks. *Artificial neural networks: methods and applications*, pp.14-22.

APPENDICES

APPENDIX A: Official Reply from Dr. Lim Siong Kang for Feedback of Objective-functions

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN
Wholly owned by UTAR Education Foundation

CHUAN MING WONG <wongm86@1utar.my>

Request for official reply for FYP

Siong Kang Lim <sklim@utar.edu.my> Thu, Sep 5, 2024 at 4:09 PM
To: CHUAN MING WONG <wongm86@1utar.my>
Cc: kckhor@utar.edu.my

Dear Chuan Ming:
After reviewing your three equations, any one of the equations is able to be used for prediction of the performance of a mixture without eggshells. However, for eggshell mixes, it still needs some adjustments to improve percentage errors within 10%.

Best Regards,
SK Lim (PhD Civil, MIEM ,PEng)
Assoc. Professor
Department of Civil Engineering
Lee Kong Chian Faculty of Engineering & Science (LKC FES)
Universiti Tunku Abdul Rahman (UTAR)
Jalan Sungai Long,
Bandar Sungai Long,
Cheras, 43000 Kajang,
Selangor Darul Ehsan.
Tel:+(603) 90860288
Fax:+(603) 90198868

[Quoted text hidden]

APPENDIX B: Results from Evaluation of Single Objective-function on Verification Dataset

V foam (theo) % of total volume	mass foam added % (of total mass)	type of mould	Age	target ρ	fresh ρ	HD	SSD D	OD D	Strength	Predicted	Difference	ABS Differ	MAE	Min	Max	Median	Std dev
32.9	1.538461538	sm	7	1300	1336	1339	1381	1135	5.15	4.95	-0.20	0.20	0.78	0.05	2.55	0.66	0.66246
32.9	1.538461538	sm	7	1300	1336	1320	1370	1127	4.50	4.81	0.31	0.31					
32.9	1.538461538	sm	7	1300	1336	1309	1361	1121	5.06	4.72	-0.34	0.34					
33.8	1.298076923	sm	7	1300	1310	1310	1353	1119	5.33	3.86	-1.47	1.47					
33.8	1.298076923	sm	7	1300	1310	1334	1375	1138	4.89	4.08	-0.81	0.81					
33.8	1.298076923	sm	7	1300	1310	1303	1345	1115	5.01	3.82	-1.19	1.19					
42.7	2.290076336	sm	7	1100	1128	1100	1159	928	2.38	2.17	-0.21	0.21					
42.7	2.290076336	sm	7	1100	1128	1132	1192	964	2.85	2.54	-0.31	0.31					
42.7	2.290076336	sm	7	1100	1128	1110	1170	939	2.37	2.28	-0.09	0.09					
43.8	2.294620138	sm	7	1100	1093	1087	1130	906	1.92	1.28	-0.64	0.64					
43.8	2.294620138	sm	7	1100	1093	1078	1123	907	1.94	1.26	-0.68	0.68					
43.8	2.294620138	sm	7	1100	1093	1082	1126	911	2.22	1.31	-0.91	0.91					
32.9	1.538461538	sm	28	1300	1336	1338	1416	1144	6.87	5.77	-1.10	1.10					
32.9	1.538461538	sm	28	1300	1336	1324	1414	1142	6.39	5.68	-0.71	0.71					
32.9	1.538461538	sm	28	1300	1336	1322	1412	1142	6.37	5.67	-0.70	0.70					
33.8	1.298076923	sm	28	1300	1310	1310	1387	1127	6.70	4.69	-2.01	2.01					
33.8	1.298076923	sm	28	1300	1310	1327	1409	1146	7.42	4.87	-2.55	2.55					
33.8	1.298076923	sm	28	1300	1310	1318	1399	1141	6.99	4.81	-2.18	2.18					
42.7	2.290076336	sm	28	1100	1128	1133	1261	973	3.19	3.24	0.05	0.05					
42.7	2.290076336	sm	28	1100	1128	1102	1235	941	2.45	2.88	0.43	0.43					
42.7	2.290076336	sm	28	1100	1128	1127	1244	962	2.79	3.18	0.39	0.39					
43.8	2.294620138	sm	28	1100	1093	1095	1190	933	3.10	2.21	-0.89	0.89					
43.8	2.294620138	sm	28	1100	1093	1092	1189	932	2.28	2.19	-0.09	0.09					
43.8	2.294620138	sm	28	1100	1093	1097	1196	933	2.77	2.20	-0.57	0.57					

APPENDIX C: Results from Evaluation of Set of Two Objective-functions on Verification Dataset

V foam (theo) % of total volume	mass foam added % (of total mass)	type of mould	Age	target ρ	fresh ρ	HD	SSD D	OD D	Strength	Predicted	Difference	ABS Differ	MAE	Min	Max	Median	Std dev
													0.80	0.03	2.50	0.62	0.62687
32.9	1.538461538	sm	7	1300	1336	1339	1381	1135	5.15	5.25	0.10	0.10					
32.9	1.538461538	sm	7	1300	1336	1320	1370	1127	4.50	5.09	0.59	0.59					
32.9	1.538461538	sm	7	1300	1336	1309	1361	1121	5.06	5.03	-0.03	0.03					
33.8	1.298076923	sm	7	1300	1310	1310	1353	1119	5.33	3.95	-1.38	1.38					
33.8	1.298076923	sm	7	1300	1310	1334	1375	1138	4.89	4.18	-0.71	0.71					
33.8	1.298076923	sm	7	1300	1310	1303	1345	1115	5.01	3.90	-1.11	1.11					
42.7	2.290076336	sm	7	1100	1128	1100	1159	928	2.38	2.74	0.36	0.36					
42.7	2.290076336	sm	7	1100	1128	1132	1192	964	2.85	3.08	0.23	0.23					
42.7	2.290076336	sm	7	1100	1128	1110	1170	939	2.37	2.86	0.49	0.49					
43.8	2.294620138	sm	7	1100	1093	1087	1130	906	1.92	1.48	-0.44	0.44					
43.8	2.294620138	sm	7	1100	1093	1078	1123	907	1.94	1.46	-0.48	0.48					
43.8	2.294620138	sm	7	1100	1093	1082	1126	911	2.22	1.48	-0.74	0.74					
32.9	1.538461538	sm	28	1300	1336	1338	1416	1144	6.87	6.01	-0.86	0.86					
32.9	1.538461538	sm	28	1300	1336	1324	1414	1142	6.39	5.98	-0.41	0.41					
32.9	1.538461538	sm	28	1300	1336	1322	1412	1142	6.37	6.00	-0.37	0.37					
33.8	1.298076923	sm	28	1300	1310	1310	1387	1127	6.70	4.74	-1.96	1.96					
33.8	1.298076923	sm	28	1300	1310	1327	1409	1146	7.42	4.92	-2.50	2.50					
33.8	1.298076923	sm	28	1300	1310	1318	1399	1141	6.99	4.87	-2.12	2.12					
42.7	2.290076336	sm	28	1100	1128	1133	1261	973	3.19	3.93	0.74	0.74					
42.7	2.290076336	sm	28	1100	1128	1102	1235	941	2.45	3.57	1.12	1.12					
42.7	2.290076336	sm	28	1100	1128	1127	1244	962	2.79	3.82	1.03	1.03					
43.8	2.294620138	sm	28	1100	1093	1095	1190	933	3.10	2.52	-0.58	0.58					
43.8	2.294620138	sm	28	1100	1093	1092	1189	932	2.28	2.41	0.13	0.13					
43.8	2.294620138	sm	28	1100	1093	1097	1196	933	2.77	2.13	-0.64	0.64					