Comparison of Different Machine Learning Algorithms to predict Mechanical Properties of Concrete

by

Bhanu Prakash koya
B. Tech, Lovely Professional University, 2017

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Mechanical Engineering

Supervisory Committee

Comparison of Different Machine Learning Algorithms to predict Mechanical Properties of Concrete

by

Bhanu Prakash koya
B. Tech, Lovely Professional University, 2017

Supervisory Committee

Caterina Valeo, Department of Mechanical Engineering
Supervisor

Rishi Gupta, Department of Civil Engineering
Co-Supervisor

Abstract

Concrete is the most widely used construction material throughout the world. Extensive experiments are conducted every year to measure various physical, mechanical, and chemical properties of concrete involving a hefty amount of money and time. This work focuses on the utilization of Machine Learning (ML) algorithms to predict a wide range of concrete properties and avoiding unnecessary experimentation. In this work, six mechanical properties of concrete namely Modulus of Rupture, Compression strength, Modulus of Elasticity, Poisson's ratio, Splitting tensile strength and Coefficient of thermal expansion were estimated by applying five different ML algorithms viz. Linear Regression, Support Vector Machine, Decision Tree, Random Forest, and Gradient Boosting models on the Wisconsin concrete mixes database. Further, these ML models were evaluated to identify the most suitable model that can reliably predict the mechanical properties of concrete. The approach followed in this research was verified using the 10-fold Cross-Validation technique to get rid of training and testing split bias. The Grid Search Cross Validation method was used to find the best hyperparameters for each algorithm. Root mean squared error (RMSE) and Nash and Suctcliffe Efficiency (NS) results showed that the Support Vector Machine outperformed all other models applied on the datasets. Support Vector Machine predicted the Modulus of Rupture at a curing age of 28 days with an NS score of 0.43 which is 34% and 26% better than the NS scores of Random Forest and Gradient Boosting advanced algorithms, respectively. This suggests that the Support Vector Machine algorithm with its NS score further improved can be used for predicting new data points at least for potentially similar systems.

Supervisory Committee

Caterina Valeo, Department of Mechanical Engineering
Supervisor

Rishi Gupta, Department of Civil Engineering
Co-Supervisor

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my sincere gratitude to my supervisors Dr. Rishi Gupta and Dr. Caterina Valeo for providing me the opportunity to study at the University of Victoria and for their supervision and support throughout my Masters program. I am grateful to Dr. Rishi Gupta for his kind advice on my research topic.

I would like to thank Dr. Sakshi Aneja, post-doctoral researcher at the University of Victoria for her generous assistance in identifying the objectives and layout of my Masters project. At last, I would like to express my gratitude to all professors, co-workers and friends who provided guidance and support during my studies.

Dedication

I dedicate this work to my parents who offered me unconditional love and encouragement. It would not have been possible to complete my Masters degree abroad without their mental and financial support. Thank you for believing in me and helping to follow my dream.

1 Introduction

1.1 Background and Motivation

Comprehensive knowledge on the properties of a material is extremely essential as they significantly influence the performance of a material and determines whether that material can be used in a particular application or not. It is very important to use materials that are very well-suited to that particular job to maximize material benefits. Generally, these properties are identified and measured by conducting experiments on a large scale using a wide range of equipment and measurement devices. These experiments would usually tend to be time-consuming, expensive and laborious. Multiple statistical and mathematical methods using empirical formulas were developed to predict these properties taking the compositions of materials as inputs [1,2]. But the issue with these statistical methods is that they are not very reliable for future predictions and are not useful when compared to computer algorithms called ML models since these models have higher accuracy than mathematical models [3].

ML is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. ML focuses on the development of computer programs that can access data and use them to learn to make future projections. ML has its applications everywhere and some of the applications in Civil Engineering include risk prediction at construction sites by image recognition, predicting the rate of change in cost of materials, and optimization of mining and construction operations etc [4,5]. One of the most valuable applications is predicting the properties of materials by ML algorithms [6]. The ability to predict material properties using its compositions or descriptors can reduce the necessity of performing physical experiments, and in turn save time and costs related to making and testing materials as well as the experimental efforts.

Concrete is the most widely used substance on the planet after water and it is the most commonly used building material worldwide. Concrete forms an essential component of a variety of structures from skyscrapers to bridges and driveways. Over 33 billion tons of concrete are being produced each year [7]. The concrete industry alone is worth over $37 billion, Unquestionably, a lot of money is spent on its production, research and experimental purposes which includes measuring its physical, mechanical, chemical properties etc. By applying ML to predict these properties we can reduce if not completely eliminate the necessity of performing physical experiments.

A lot of work was done to predict compressive strength which is the most important property of concrete using different ML algorithms such as Decision Tree, Support Vector Machine as well as Artificial Neural Networks [8,9]. But little research was performed applying ML models to predict mechanical properties such as Modulus of Elasticity, Modulus of Rupture, Coefficient of thermal expansion etc and Gradient Boosting algorithm was never used for property prediction. Predicting the compressive strength of concrete is relatively easier than predicting these mechanical properties since the relation between concrete components and these properties is highly non-linear. Therefore, there is a need to go for more sophisticated and unconventional algorithms for property prediction [10]. Ensemble methods is a technique in which a complex algorithm is built

by combing many individual models. They are efficient ML tools to solve very complex problems that cannot be solved by individual ML models and there were fewer studies to predict properties using ensemble technique. So, this work focuses on applying basic to advanced ML algorithms including the Gradient Boosting ensemble model to predict these mechanical properties of concrete.

1.2 Thesis Outline

This thesis is presented in chapters. Chapter 1 briefly describes the purpose, scope of this study as well as the objectives of this project. Chapters 2 gives an explanation about ML and its life cycle. Chapter 3 describes the different ML Algorithms applied in this project and explains the procedures involved in the ML life cycle with emphasis on the work done on the collected datasets to achieve the objectives.

Chapter 4 explains the obtained results by comparing all the models applied to predict 7 days Modulus of Rupture and different properties of concrete and discusses the reasons behinds the results to better understand characteristics of ML models to forecast material's properties. Chapter 5 provides the conclusions drawn from this study and the possible future work to extend this study.

1.3 Objectives

The primary objective of this project is to predict different properties of concrete by using ML algorithms and to identify the most suitable ML algorithms so that they can be used for future reliable predictions. Properties of concrete include Modulus of Rupture, Compression strength, Modulus of Elasticity, Poisson's ratio, Splitting tensile strength and Coefficient of thermal expansion

The objectives of this work are summarized as follows:

1. Perform necessary data cleaning on the concrete datasets collected from different databases to remove any corrupted data points.
2. Perform Exploratory Data Analysis on the collected data points to find the relation between input and output variables as well as the correlation among input features in the dataset.
3. Apply various individual ML algorithms and ensemble models on the collected datasets and tune their hyperparameters to best suit the datasets.
4. Evaluate the performance of the ML models based on the RMSE values and NS score to determine the best algorithm to predict the concrete properties.
5. Determine the most important features i.e, constituent materials in concrete from the input variables to predict the properties.

2 Machine Learning

2.1 Definition

ML is a branch of Artificial Intelligence that helps to train computers to learn from data through experience. With the recent advancements in computational power and the availability of a huge amount of data. computers can be trained to perform a basic task of house price prediction to advanced tasks such as image classification, voice recognition, Natural Language Processing, etc. This enabled the implementation of ML in a variety of applications such as autonomous driving, language translation and even in scientific research [11]. The process of making computers learn from data using ML algorithms is known as training. Based on the training method and the data fed into the models, ML algorithms can be classified as supervised and unsupervised machine learning.

In supervised ML, data is given to the model with the respective output label. The purpose of training is to find an appropriate mathematical function that can map the given input to the respective output so that the model can be used to predict output with high accuracy for data that were not used to train that ML model. The performance of a trained ML model is evaluated using its prediction on test data, which are data that were not used for training and the output is known. A schematic diagram representing supervised ML is shown in Figure 1. The input data given to an ML model are called "features" or "Input variables" and the output is called "predictor". Supervised ML is divided into two categories: Regression in which the prediction is a continuous variable and Classification in which the output is categorized into different classes. There are many supervised ML algorithms available such as Linear Regression, Logistic Regression, Decision Tree, and Artificial Neural Networks (ANN), etc [12].

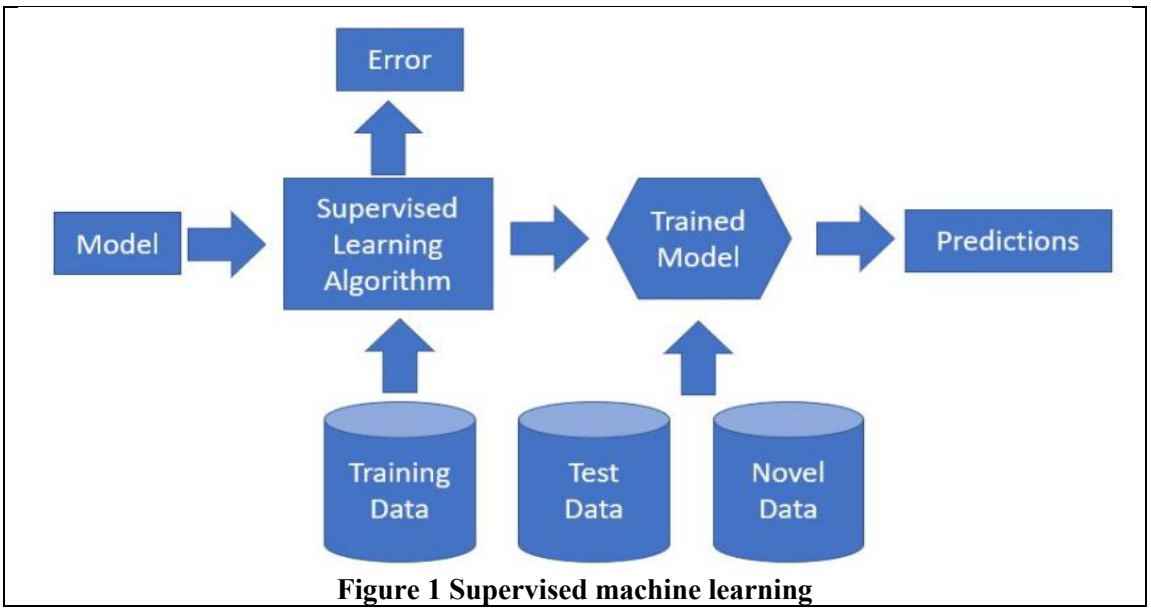

Figure 1 Supervised machine learning

In an unsupervised ML, also known as clustering, only training data are given to the ML model. The output of the data is unknown. Clustering algorithms learn from the data, find similarities and

group data into different clusters such that samples within a single cluster are as similar as possible whereas samples in different clusters as different as possible [13]. The objective function in unsupervised learning is often the sum of squared distance between all samples within the same cluster. The optimization aims at minimizing the sum of the squared error across all clusters. Some examples of Unsupervised ML algorithms are K-Means algorithm, Agglomerative clustering, Fuzzy C-Means, algorithm. A schematic diagram representing unsupervised ML is shown in Figure 2.

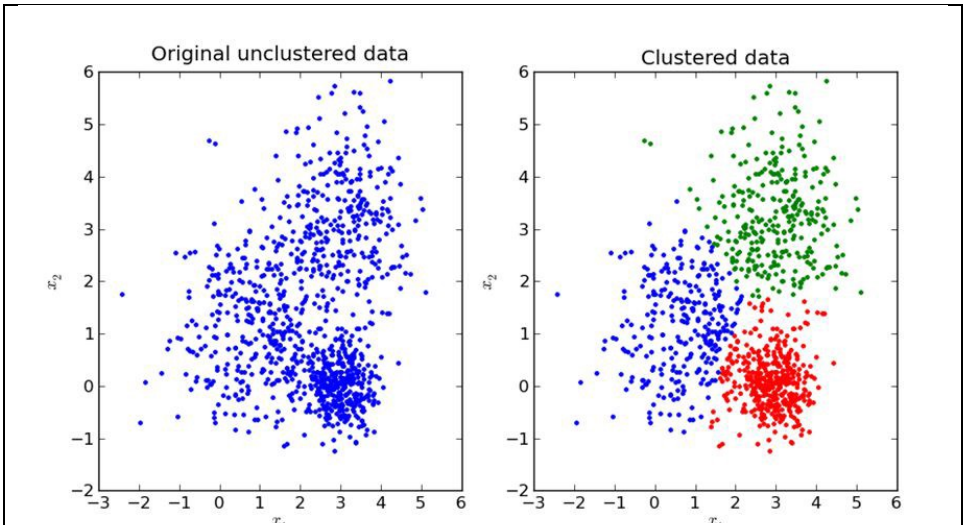

Figure 2 Unsupervised machine learning

2.2 Machine Learning Life Cycle

ML system works based on a Life cycle which is a cyclic process to find a solution for a specific problem. This cyclic process involves seven major steps as shown in Figure 3. The most basic thing in the whole process is to understand the problem and to know the purpose of the problem. So, before starting the life cycle, there should be a complete understanding of the problem since obtaining a good result depends on a better understanding of it. In the process of solving that problem, an ML system called "model" is created by training it. But to train a model we need data. So, the ML life cycle starts with collecting data. All these processes are explained in the coming sections with reference to the work done in this project.

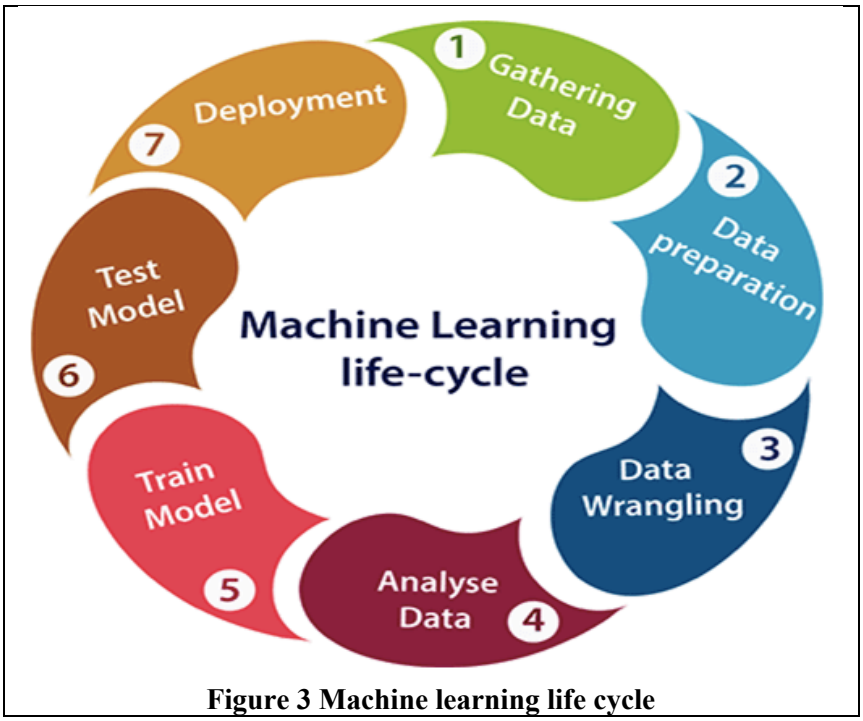

Figure 3 Machine learning life cycle

3 Methodology

3.1 Data Abstraction or Gathering

Data Gathering is the first step of the ML life cycle. The purpose of this step is to identify and collect all the data related to the problem we want to solve. In this stage, all the different data sources such as databases, document files, internet are identified. This is one of the vital stages in the life cycle since the quality and quantity of the data gathered determines the efficiency of the results. The more accurate the data, the more accurate the prediction. This step comprises the tasks of identifying various data sources, collect data and integrate all the data acquired. The end result of all these steps is a coherent set of data called "Dataset" which will be used in further steps. Now the data gathering process with respect to the data collected in this project is explained in detail.

The data used in this study was extracted from the AASHTO Mechanistic Empirical Pavement Design Guide (MEPDG) study [14] that was performed to find the effects of different concrete constituent materials on its key mechanical and thermal properties of concrete. It contains samples of concrete with varied mix proportions of constituents that were used to measure the effects on a total of six mechanical properties of concrete which are Modulus of Rupture, Compression strength, Modulus of Elasticity, Poisson's ratio, Splitting tensile strength and Coefficient of thermal expansion. The six mechanical properties, also called predictors or output features were predicted from a total of 10 input features which are Mix Types, Cement Source, Supplementary Cementitious Material, Fine Aggregate Type, Coarse Aggregate Type, Air Entraining Admixture, Air Content, Slump, Water to Cement Ratio, Water Reducing Admixture. Coefficient of thermal expansion was measured at an age of 28 days and all the other five mechanical properties were measured at ages of 7, 14, 28, and 90 days making the total number of predicted properties to 21.

The Input features Mix types, Cement source, Supplementary cementitious material, Fine aggregate type, coarse aggregate type in the dataset are categorical data types which means each of these features does not have any correlation between them and the rest of the input features are continuous data types. These ten input features will actually become 17 features when these categorical variables are converted to numbers since ML models work only with numbers. Now detailed description of the mix proportions and properties of constituent materials is provided below.

The categorical input features Cement Source has two groups which are Type I ordinary Portland cement from two different sources, Supplementary Cementitious Material has Slag, Fly Ash and None as three categories in it, Coarse Aggregate Type contains glacial gravels and crushed stone as two major groups, Fine Aggregate Type has Sand-A and Sand-B as two different classes and Mix Types contains Grade A, Grade A-S, and Grade A-F as three categories which are the mix design types found from Wisconsin Department of Transportation (WisDOT) [15]. The continuous input features Air Entraining Admixture ranges from 7 to 30ml/2.5 ft^3, Air Content varies from 3.4% to 6.8%, Slump has a minimum value of 1m and maximum value of 3m, Water to Cement Ratio has a range of 0.33 to 0.4 and Water Reducing Admixture ranges from 0 to 125ml/2.5 ft^3. This data was adopted from [3]. From these samples of different mix proportions, the mechanical properties were measured and can be used in ML for approximating properties of future mixes.

There were two batches produced for each mix. The first batch is denoted as batch A and its specimens were used for modulus of rupture and coefficient of thermal expansion measurements. Compression strength, modulus of elasticity, Poisson's ratio, and splitting tensile strength were measured on the specimens of the second batch denoted as batch B. Mix proportions were varied to obtain 110 different samples. The input data for the five categorical features named mix types, cement source, supplementary cementitious material, fine aggregate type and coarse aggregate type was collected from the four mix matrices referenced [14].

3.2 Data preparation

After the data is collected, it should be prepared for further steps. In this process of Data preparation, the data are kept in a suitable place to preparation for use in the ML training. In this step, all the data are put together, randomize the ordering if needed and explore the data which we are working with to understand its characteristics, structure and quality. The data preparation procedure followed on the collected data is explained now.

Since two batches were used to measure different properties of concrete as mentioned above, two spreadsheets named "Dataset_A", "Dataset_B" were created to build the dataset to feed into the ML models to predict the mentioned properties. The data on which the output variables (i.e., properties we are trying to find out) are dependent on and are determined by are called Independent features and the features which are being predicted are called dependent features. The independent categorical features data for all the measured properties are the same which means both the spreadsheets have the same data for the categorical features. So, first in both the spreadsheets a column named 'Mix No' that contains 110 rows with consecutive numbers 1 to 110 each row representing a mix was formed. Next, five columns named "Mix types", "Cement source", "Supplementary cementitious material", "Fine aggregate type", "coarse aggregate type" were

formed and the data for the listed five categorical input features was acquired from the four Mix Matrices by filling up each mix's data in its corresponding row in both the spreadsheets as the data for the categorical features is same for all the properties. The top ten rows filled up with categorical data in the spreadsheet is shown in Figure 4.

Mix No	Mix Types	Cement Source	Supplementary Cementitious Material (SCM)	Fine Aggregate Type	Coarse Aggregate
1	A	Source 1	None	Sand A (igneous)	Glacial Granite
2	A	Source 1	None	Sand A (igneous)	Glacial Granite
3	A	Source 1	None	Sand A (igneous)	Glacial Granite
4	A	Source 1	None	Sand A (igneous)	Glacial Granite
5	A	Source 1	None	Sand A (igneous)	Glacial Granite
6	A	Source 1	None	Sand A (igneous)	Glacial Granite
7	A	Source 1	None	Sand A (igneous)	Crushed Stone
8	A	Source 1	None	Sand A (igneous)	Crushed Stone
9	A	Source 1	None	Sand A (igneous)	Crushed Stone
10	A	Source 1	None	Sand A (igneous)	Crushed Stone

Figure 4 Independent categorical features data

The "Dataset_A" denotes that information contained in it was from the batch A specimens which were used to measure 7, 14, 28, 90 days modulus of rupture and coefficient of thermal expansion measured after 28 days. Five columns named "Air entraining admixture", "Air content", "Slump", "Water to cement ratio" and "Water reducing admixture" representing the five numerical input features were added and filled with corresponding mix data taken from the Fresh concrete properties mix matrices in [14] taking only the required numerical columns and rows where Batch ID is "A" and the associated top nine rows from the spreadsheet is shown in Figure 5.

1	Air Entraining Admixture (AEA) (mL)	Air Content (%)	Slump (in)	Water-to-Cement Ratio (WCM)	Water Reducing Admixture (WRA) (mL)
2	12	6	2.25	0.4	50
3	11	6.2	2.75	0.4	65
4	12	4.3	1.25	0.4	60
5	11	5.7	2.75	0.4	60
6	9	5.1	1.5	0.4	120
7	12	5.3	2.5	0.4	60
8	10	5.6	3	0.4	60
9	18	5.8	2.25	0.4	85
10	15	5.5	1.25	0.4	85

Figure 5 Independent numerical features data

These ten columns added so far was the independent features data for the machine learning models. Another five columns named "Modulus of Rupture (7 days)", "Modulus of Rupture (14 days)", "Modulus of Rupture (28 days)", "Modulus of Rupture (90 days)" and CTE (28 days) representing the output features were created and filled with corresponding mix data from the measured modulus of rupture for different specified days and coefficient of thermal expansion after curing for 28 days with columns taken from the tables in Appendix X of referenced [14] and the top nine rows is shown in Figure 6. This formed the complete dataset (Dataset_A) for the ML algorithms and the dataset can be found here at Dataset_A.

1	Modulus of Rupture (PSI) (7 Days)	Modulus of Rupture (PSI) (14 Days)	Modulus of Rupture (PSI) (28 Days)	Modulus of Rupture (PSI) (90 Days)	CTE (mm/mm/^{0}C) (28 Days)
2	606	684	676	817	0.000009341
3	623	605	675	829	0.000009776
4	644	665	758	816	0.000009183
5	613	630	682	734	0.000008671
6	631	688	782	904	0.000009577
7	612	613	666	804	0.000009503
8	672	646	719	772	0.000009527
9	674	702	765	721	0.000008492
10	680	702	785	866	0.000009324

Figure 6 Dependent features data

Now, the data collection process for the Dataset_B is discussed. The five categorical columns were already filled up and only numerical features data needed to be collected. Five columns which were the same as in Dataset_A named "Air entraining admixture", "Air content", "Slump", "Water to cement ratio" and "Water reducing admixture" representing the five numerical input features were added and filled with corresponding mix data taken from the Fresh concrete properties mix matrices in [14] taking only the required numerical columns and rows where Batch ID is "B". These ten columns added so far were the independent features data for the machine learning models to predict compression strength, modulus of elasticity, Poisson's ratio, and splitting tensile strength for different curing ages. A total of 16 columns i.e. four columns for compression strength, modulus of elasticity, Poisson's ratio, and splitting tensile strength were added for curing ages of 7, 14, 28, 90 days for each of them representing the output features were created and filled with corresponding mix data from the measured values taken from the tables in Appendix X of reference [14] and the top nine rows for compression strength are shown in Figure 7. This formed the complete dataset (Dataset_B) for the ML algorithms and the dataset can be downloaded from Dataset_B.

1	Compressive Strength (PSI) (7 days)	Compressive Strength (PSI) (14 days)	Compressive Strength (PSI) (28 days)	Compressive Strength (PSI) (90 days)
2	4365	4878	5307	6166
3	3734	4350	4686	5475
4	3764	4012	4433	4898
5	4319	4657	5042	5727
6	5054	5852	6081	6770
7	3553	4041	4472	5088
8	4013	4504	4855	5620
9	4676	5189	5738	6524
10	5070	5621	6337	6758

Figure 7 Compression strength data

Thus, the datasets for predicting different mechanical properties were collected. Since many of the properties were measured at multiple curing ages each curing age was treated separately for each property such that four predictions were made for compression strength, modulus of elasticity, modulus of rupture, splitting tensile strength, and Poisson's ratio corresponding to the four curing ages. All the coming processes were applied on both Dataset_A and Dataset_B, with detail explanation given on Dataset_A.

3.3 Data Wrangling or Pre-processing

The next step in the life cycle is called "Data wrangling" which is the process of cleaning and converting the raw data into a usable format. It is the process of cleaning up the data, choosing the variables to use, and transforming the data into an appropriate format to make it more suitable for analysis in the next steps. The real-world data collected may not always be readily useful and may have various issues such as repeated data, missing values, incorrect data, etc. So, using various techniques the collected data will be cleaned. It is mandatory to detect and remove the mentioned issues because it can adversely affect the quality of the outcome. Cleaning the data is essential to deal with the quality issues so, it is one of the important steps in the complete process. The Data pre-processing done on the Dataset_A is explained step by step below.

Python is the programming language used to write the code in this project. It is one of the most used coding languages with applications in a wide range of fields such as web development, Machine Learning, etc. We need an interface to run code in any language called code editors or integrated development environment (IDE). Google Colaboratory is an IDE used to write and execute python code in the browser without needing to install any software and it is the IDE used in this project.

No code was written or run until this point and everything from now on was achieved by code written in python. First, the Dataset_A which was in the spreadsheet with the name "Dataset_A.xlsx" was uploaded into the Google Colaboratory Notebook. The dataset was read as a data frame which is the primary data structure of pandas [16]. It consists of the data organized in rows which run horizontally and columns which run vertically. The dataset has 16 columns representing all the 10 input features, 5 dependent features with the first column being Mix number and 110 rows-each row representing all the information for each sample.

The Data cleaning in this project involved checking the data types of all the feature columns and finding Null values if any exist. All the feature columns as well as the outputs are all integer and float types except the categorical input features which are object data type in python. If the data types of any of the columns were misinterpreted by the IDE they should be changed into the desired type of data. The next step was to check any missing values in the data, also called Null values and if there are any missing values, that particular row cannot be used for training and that particular row should be dropped altogether or the missing data point should be replaced by mean or median strategy. In real-world datasets, there would always be a few missing values. In python, null is represented as NaN and these two words can be interchangeably used. However, removing the rows from the dataset is not a good option as the valuable information could be lost. Dropping 4 or 5 rows from a dataset containing a hundred thousand data points may not affect the results greatly whereas dropping 5 rows if the dataset has only 150 data points is not a good idea. If the missing data point is numerical, it can be replaced with the mean of all the values of that particular column and if the missing data point is categorical, it should be replaced by the most repeated value in that column called "Median" since we can't take average of categorical values (strings). Figure 8 shows the datatypes as well as the number of missing values in all the data columns and it was clear that none of the columns had missing values.

#	Column	Non-Null Count	Dtype
0	Mix No	110 non-null	int64
1	Mix Types	110 non-null	object
2	Cement Source	110 non-null	object
3	Supplementary Cementitious Material (SCM)	110 non-null	object
4	Fine Aggregate Type	110 non-null	object
5	Coarse Aggregate	110 non-null	object
6	Air Entraining Admixture (AEA) (mL)	110 non-null	float64
7	Air Content (%)	110 non-null	float64
8	Slump (in)	110 non-null	float64
9	Water-to-Cement Ratio (WCM)	110 non-null	float64
10	Water Reducing Admixture (WRA) (mL)	110 non-null	int64
11	Modulus of Rupture (PSI) (7 Days)	110 non-null	int64
12	Modulus of Rupture (PSI) (14 Days)	110 non-null	int64
13	Modulus of Rupture (PSI) (28 Days)	110 non-null	int64
14	Modulus of Rupture (PSI) (90 Days)	110 non-null	int64
15	CTE (mm/mm/0C) (28 Days)	110 non-null	float64

Figure 8 Data types of Input and output feature columns

3.4 Data Analysis

Now the cleaned data is ready to be analyzed. Data Analysis also referred to as data analytics or Exploratory data analysis (EDA) is a process of finding patterns in the data, understanding it and try to obtain inferences so that the underlying patterns can be observed. In this process, the cleaned data is inspected, and necessary transformations are performed in order to discover useful information, derive conclusions, and to support decision-making in a way humans can understand and decide better. The analysis performed on the pre-processed dataset to gain insights is explained below.

3.4.1 Measures of central tendency

First, eight statistical properties of the data which are count, mean, standard deviation, minimum, 25%, 50%, 75% and maximum values of each of the numerical columns were examined. Note that the first column is mix number that contains consecutive numbers from 1 to 110 and does not provide any insights on the data. This was done to check and see how the data is distributed and all the values of these eight properties of both the independent and dependant numerical variables can be seen in

Figure 9. The 25%, 50%, 75% values represent the value of 25^{th} %, 50^{th} %, 75^{th} % data point respectively when the data is arranged in ascending order. “Std” is the Standard deviation of a variable which is the measure of the amount of variation of the set of values. It shows how far the values are dispersed from the mean and a low standard deviation implies that the values tend to be closer to the mean of that set, whereas a high standard deviation indicates that the values are spread out over a wider range. This is the most important of all the metrics explored here since standard deviation can be used to evaluate the performance of machine learning models which will be shown in the coming sections.

	Mix No	Air Entraining Admixture (AEA) (mL)	Air Content (%)	Slump (in)	Water-to-Cement Ratio (WCM)	Water Reducing Admixture (WRA) (mL)	Modulus of Rupture (PSI) (7 Days)	Modulus of Rupture (PSI) (14 Days)	Modulus of Rupture (PSI) (28 Days)	Modulus of Rupture (PSI) (90 Days)	CTE (mm/mm/0C) (28 Days)
count	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	1.100000e+02
mean	55.500000	16.431818	5.240909	2.413636	0.387636	31.336364	662.809091	719.190909	781.545455	874.672727	9.472336e-06
std	31.898276	4.839856	0.674475	0.561000	0.017916	37.958277	65.716586	76.224619	72.508606	71.980516	4.505294e-07
min	1.000000	8.000000	3.400000	1.000000	0.330000	0.000000	508.000000	575.000000	623.000000	721.000000	8.492000e-06
25%	28.250000	12.000000	4.725000	2.000000	0.380000	0.000000	613.000000	663.500000	727.250000	826.750000	9.334000e-06
50%	55.500000	16.000000	5.300000	2.500000	0.400000	10.000000	662.000000	706.500000	781.500000	877.000000	9.508000e-06
75%	82.750000	20.000000	5.775000	3.000000	0.400000	60.000000	703.000000	769.500000	824.750000	922.000000	9.674250e-06
max	110.000000	31.000000	6.600000	3.000000	0.400000	125.000000	849.000000	942.000000	972.000000	1080.000000	1.098400e-05

Figure 9 Statistical properties of features

This step cannot be performed on the categorical features since all these eight functions need numerical data. In order to gain insights on the categorical features, histograms were generated to find out the most repeated classes in the categorical features data columns which can be seen in Figure 10. The longest bin in each histogram shows the most repeated class in each of the categorical feature data columns.

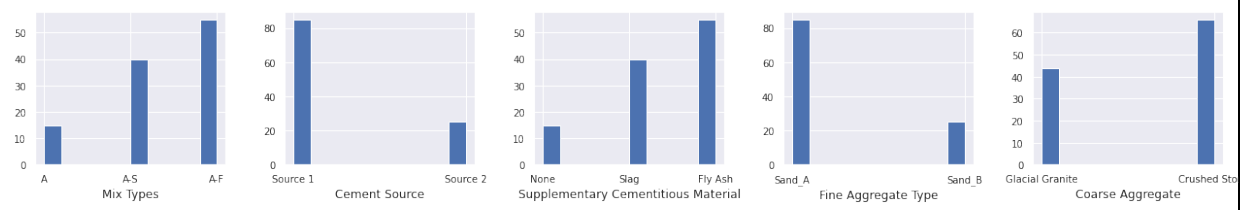

Figure 10 Histograms of categorical feature columns

3.4.2 Outlier Removal

The next step was to remove outliers in the dataset. An outlier is a data point in any of the columns that is significantly different from the other observations of that specific column [17]. Outliers which are the most extreme observations occur in the data due to a lot of factors such as experimental error, measuring error or mishandling of the data while storing and moving from one database to another. They indicate faulty data and what might be the reason behind outlier occurrence, they must be removed from the dataset.

One of the strategies of removing outliers and the one used in this project is based on quartile ranges. As an example, the process of outlier removal is shown in the feature column "Water to cement Ratio". To graphically show outliers, a box plot was created as shown in Figure 11. Box plot is also known as whisker plot is a kind of chart used in data analysis to visually show the distribution of numerical data by displaying the data quartiles or percentiles. It shows the summary of a dataset at 5 quartiles which are minimum value, first (lower) quartile, median, third (upper) quartile, and maximum value.

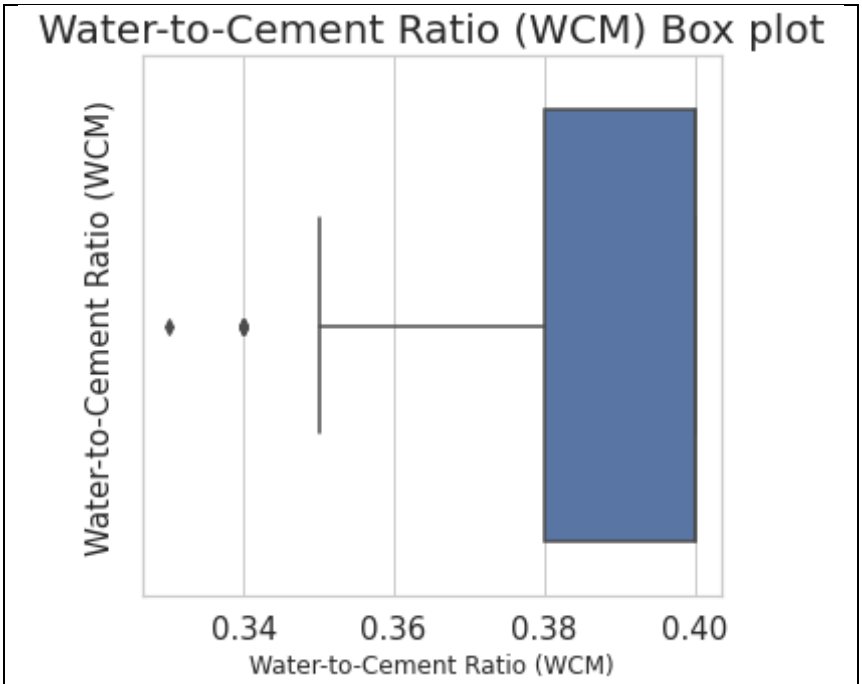

Figure 11 Box plot

As seen in the box plot, the water to cement ratio is widely spread from a minimum of 0.325 to a maximum of 0.40. The vertical lines on the right and left side of the blue box are called "whiskers", and the diamonds in the box plot represent outliers in the data. All data points that are less than the (Q1-1.5*IQR) in the box plot and that are higher than the (Q3+1.5*IQR) are considered as outliers [18]. In which Q1 is the first quartile of the distribution which is the 25th % data point, Q3 is the third quartile of the distribution which is the 75th % data point and IQR is the Inter Quartile Range i.e., Q3-Q1.

There are 7 outliers in the "water to cement ratio" in total. In the same way outliers in all the feature columns were detected and it was discovered that only "water to cement ratio" feature column has outliers. There are eight unique values in water to cement ratio feature column and these values are just the fixed amounts of water per unit of cement present in each sample mix. So, applying this strategy and removing the outliers is not a good idea here and also due to the fact that this dataset has only 110 rows i.e., 110 samples, the rows containing the outliers were not dropped since there was a chance of losing the valuable information from those data samples.

3.4.3 Multi collinearity Correlation Matrix

The next step is to check the correlation among the features. Correlation or collinearity is how much one variable depends on other variables i.e, how much a variable varies with change in another variable. Multicollinearity in a dataset occurs when there are features that are strongly dependent on each other i.e, increasing one variable increases the other variable, increasing one variable decreases the values of the other and vice versa. The existence of multicollinearity in the input feature columns negatively impacts the performance of ML models as it impacts the interpretability of the models. When an output variable can be predicted from features that are

inter-dependant on each other, they can be dropped from the dataset to reduce the complexity of the model as well the time taken for the model to run will also be decreased.

Now the process of finding out multicollinearity in our dataset is discussed. The simplest method to identify it to plot a pair plot between the features which is nothing but a 2-dimensional graph between the selected two features. Pair plots just show how two features vary with respect to each another graphically, but they do not provide a clear thought on whether to keep the features in the dataset or drop them and to decide this a numerical correlation value is required. There are many correlation coefficients used to measure how strong a relationship is between two variables but the most popular one out of all is the Pearson correlation coefficient. It shows the relationship between two features by giving a numerical value. It is represented by ρ and it can either be positive or negative.

$$\rho = \frac{\sum_{i=0}^{n}(x_i - x_m)(y_i - y_m)}{\sigma_x * \sigma_y} \qquad (3.1)$$

where x_i, y_i are each data points in *x* and *y* feature columns and x_m, y_m, σ_x, σ_y are the mean and standard deviation values of *x* and *y* feature columns, respectively.

Correlation coefficient of 1 means for every unit increase in one variable, there is an increase of a fixed proportion in the other variable. A correlation coefficient of -1 means that for every unit increase in one variable, there is a decrease in a fixed proportion in the other. Zero means that for every increase or decrease, there isn't an increase or decrease which means that two variables just aren't related. The absolute value of correlation coefficient gives the relationship strength and the larger the number is, the stronger the relationship would be. For example, the strength of correlation coefficient of -0.95 is 0.95, which has a stronger relationship than 0.94 and under.

Now, the Pearson correlation coefficient of each feature with every other feature column including the output variables was found and is shown in Figure 12. This gives a much better understanding of the correlation between features in the form of a numerical value. Each row in the correlation matrix represents a feature and its correlation with all the other features are shown in the columns of that particular row and the color variation in the color bar of this heat map shows the strength of correlation. Notice the first column is Mix number and gives no meaningful inference.

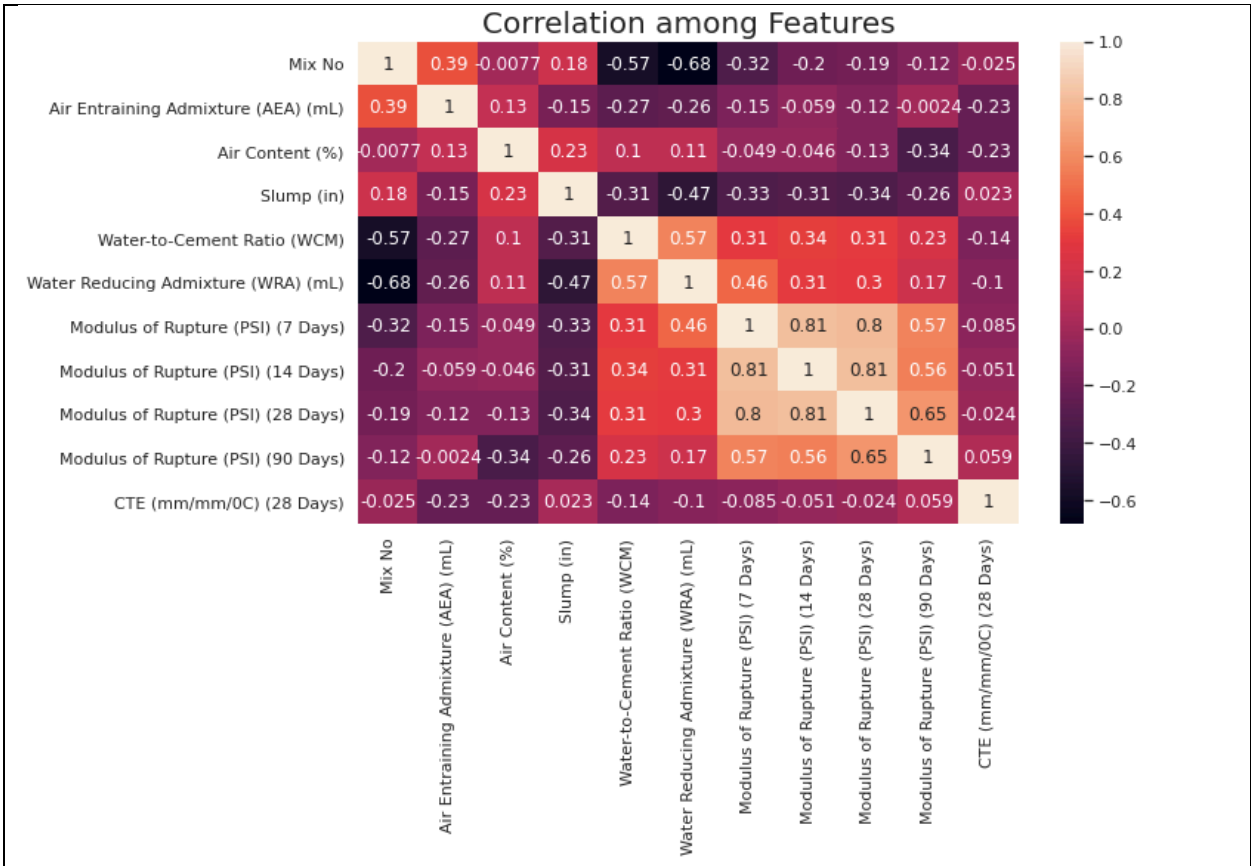

Figure 12 Correlation matrix

The below inferences can be drawn from the correlation matrix.

- In the input features, water reducing admixture has a high correlation of 0.57 with water to cement ratio and -0.47 with slump which are significant.
- The output feature Modulus of rupture has a negative correlation with slump and good positive correlation with water to cement ratio and water reducing admixture.
- There are no high correlations between CTE and any input features.

We know that some of the features should be dropped from the dataset if there is multi collinearity existing in the dataset. But there is a threshold value for which the features can be dropped if they exceed that. It is usually taken as 0.8 and this threshold value will change depending upon the use case [19,20]. Since there are no correlations among features which are higher than 0.8, it is decided to not to drop any of the input features to feed the ML models. The correlation can further be analysed by plotting some interesting visualizations between the features and some useful interpretations can be obtained as described in the next section.

3.4.4 Visualizations

To gain in depth understanding of interrelations among the input features as well as between input and output features, some scatter plots were generated between them and described along with the insights gained. For all the scatter plots shown below the output feature is taken on Y-axis, input

feature on X-axis, the varying size and colour of the scatter plot markers represent two other input features with colour changing between lighter and darker shades.

i. Modulus of Rupture (28 days) versus Air Entraining Admixture with varying Water to Cement Ratio and Air Content.

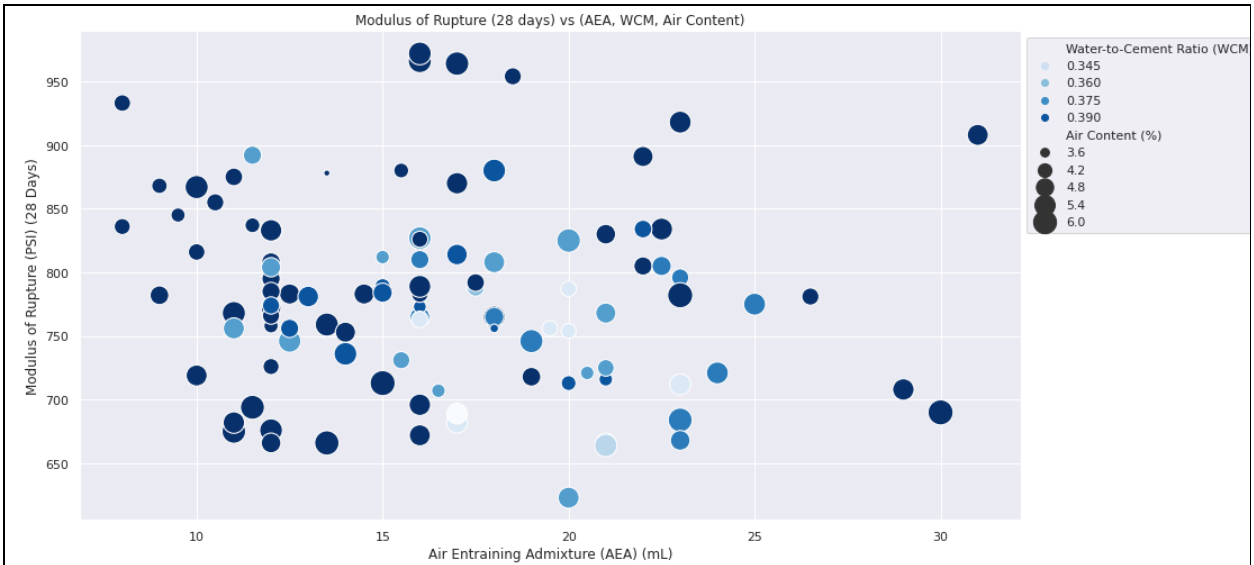

Figure 13 Modulus of Rupture (28 days) vs (AEA, WCM, Air Content)

The below interpretations were drawn from this visualization.

- Water to Cement Ratio seems to be increasing as Air Content increases because darker dots appear larger in size.
- Modulus of Rupture appears to be increasing with Water to Cement Ratio and can be noticed from the high number of darker points lying at the upper area of graph.

ii. Modulus of Rupture (28 days) versus Air Entraining Admixture with varying Slump and Air Content.

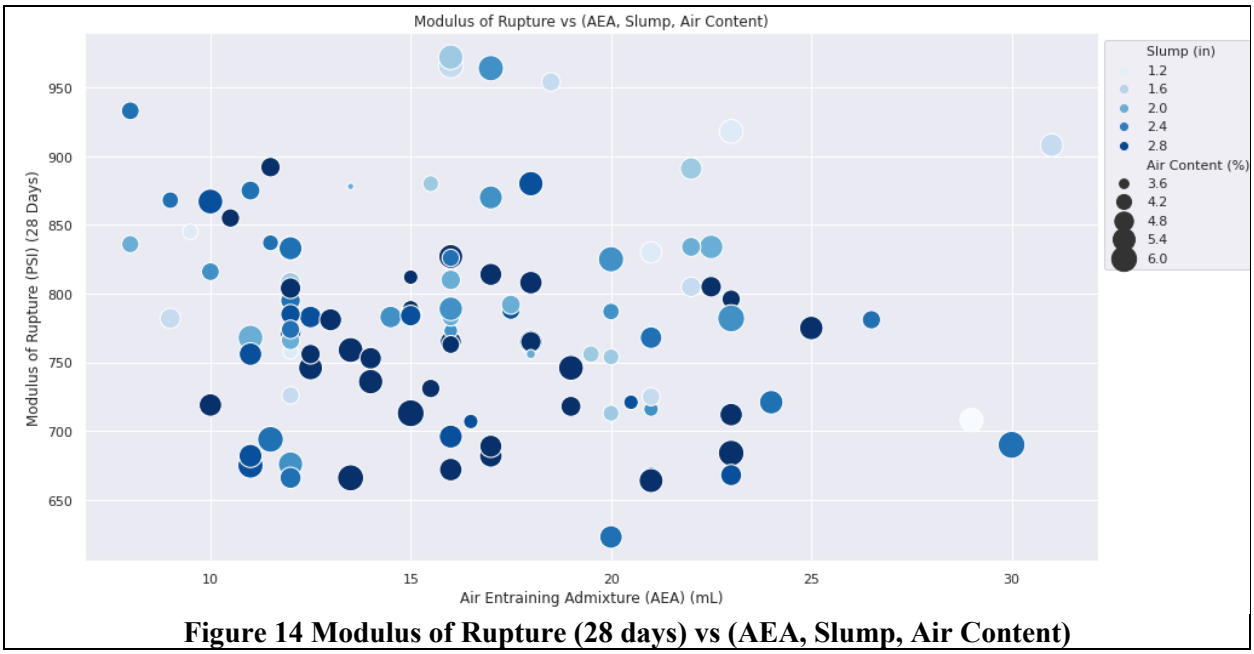

Figure 14 Modulus of Rupture (28 days) vs (AEA, Slump, Air Content)

The graph gives the insights explained below.

- Modulus of Rupture looks increasing as Slump decreases, and this can be noticed from the lighter markers present at the higher Modulus of Rupture side.
- The more Slump the mix is, the more is the Air Content it requires for low Modulus of Rupture as large darker points are concentrated at the lower left part of the graph i.e, at low Modulus of Rupture.

All these plots were generated by plotting Modulus of Rupture on Y-axis taken at 28 days. But, the same inferences can also be projected to all the other 7, 14 and 90 days Modulus of Rupture variables since the correlation among all the different days Modulus of Rupture columns is very high at 0.81 as shown in the correlation matrix Figure 12 (above).

iii. Coefficient of thermal expansion (28 days) versus Air Entraining Admixture with varying Water reducing Admixture and Air Content.

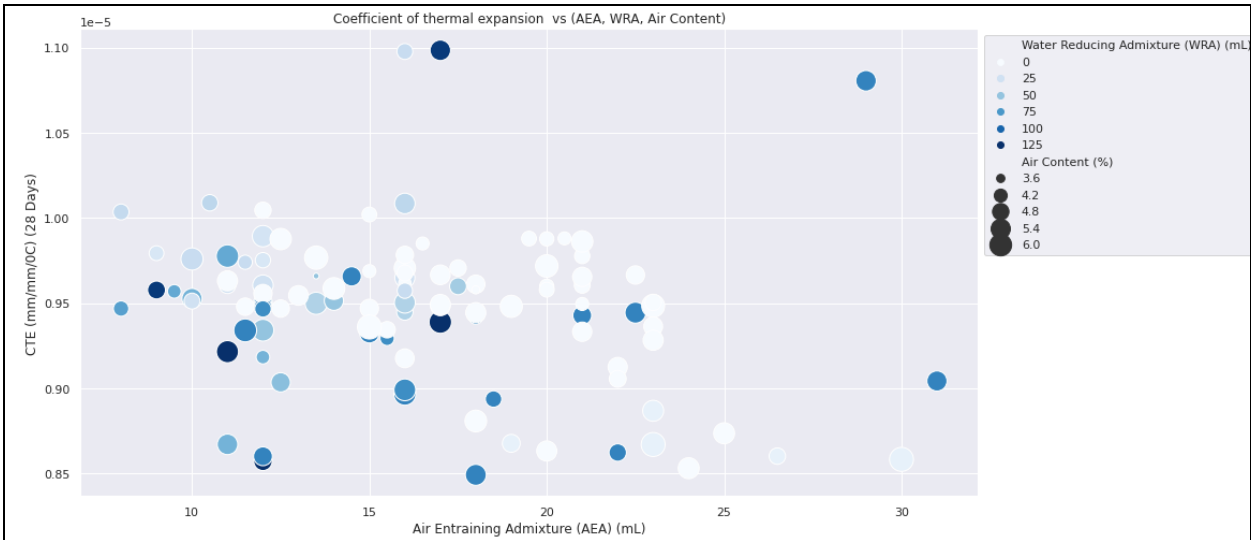

Figure 15 Coefficient of thermal expansion (28 days) vs (AEA, WRA, Air Content)

Now, the other output variable Coefficient of thermal expansion measured at 28 days is taken on Y-axis and plotted against different input feature columns and the below insights were observed.

- CTE seems to be decreasing on increasing the Air Entraining Admixture amount in the mix as the markers on the lower right side of plot with high Air Entraining Admixture values has less Coefficient of thermal expansion.
- There seems to be no linear relation between CTE and Water reducing Admixture.

iv. Coefficient of thermal expansion (28 days) versus Slump with varying Air Entraining Admixture and Water to Cement Ratio.

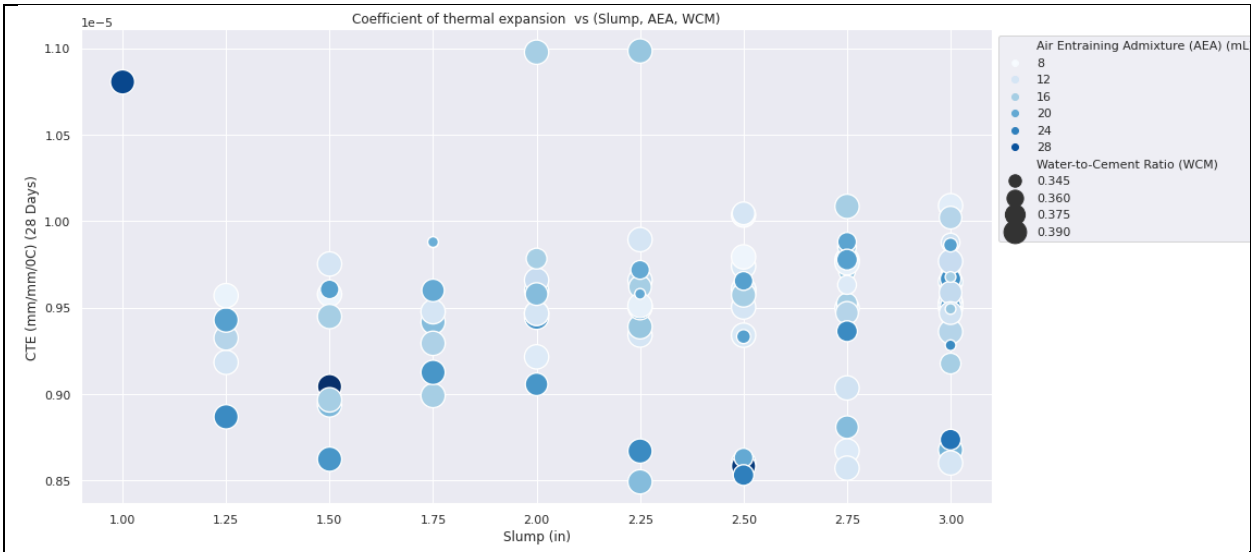

Figure 16 Coefficient of thermal expansion (28 days) vs (Slump, AEA, WCM)

From the above plot generated the below insights were drawn.

- It looks the more the Slump in the mix is, the less is the Water to Cement Ratio for maintaining good CTE as smaller points are concentrated at the right side of the graph i.e, at high Slump.
- Coefficient of thermal expansion is high when Water to Cement Ratio is high and requires less Air Entraining Admixture.

The data analysis part on the dataset is completed and some of the useful insights were extracted. Though conclusions were made by observing the scatter plots generated, there is an underlying non-linear interaction between the features which cannot be visualized by any complex graphs. We can visually understand 2D, 3D and maximum up to 4-demensional plots, here 4D meant the color and size variations represented by features as shown above. We can keep on producing complex and fancy graphs and do further analysis by doing row wise and column wise feature plotting but still we lack the ability to track all these correlations by ourselves. This is where ML comes into picture where these underlying relations will be captured to give better insights into the problem.

Now, we need to perform some operations on the features in the dataset to be able to affectively use them in the machine learning. From here on we will start processing the data to be fed into the machine learning models to correctly predict the mechanical properties given the input features.

3.4.5 Handling Categorical Variables

As already mentioned, the categorical features existing in the dataset need to be converted into numbers since the ML models will accept and deal only with numbers, here the process of converting these categorical variables into numbers is discussed. As we know that the categorical data is of two types: Nominal data and Ordinal data. It is important to note that the process for converting the Ordinal data is different from that of nominal data. Ordinal data is processed by assigning a numerical value to all the unique groups available in that particular feature column. For example, a store experience data column has three unique groups in it which are bad, good, excellent. Then, each of the groups are a assigned a number such as 1 for bad, 2 for good and 3 for excellent and this method is called “Label Encoding”. The machine learning models assumes some sort of relationship between these variables which is correct and gives results accordingly [21].

The same method should not be followed to process Nominal variables. All the categorical features in our dataset are Nominal types and the process to convert them into numerical data is now explained. Consider the Supplementary Cementitious Material feature column from the dataset having Nominal categorical data containing three groups of the type of cementitious material used which are Fly Ash, Slag and None. If Label Encoding is applied here and the groups are assigned a number such as 1 for Fly Ash, 2 for Slag, 3 for None, the ML models will form some relations like (Fly Ash < Slag < None) or (Fly Ash + Slag = None) which make no sense. Though the ML models give the results using this method they will not be optimal. So, a technique called One-Hot Encoding is used to process this feature column. In One-Hot Encoding what we basically do is to create ’n’ columns where n is the number of unique values that the nominal variable has. Here SCM feature has 3 unique groups then three new columns namely SCM_Fly Ash, SCM_Slag, SCM_None were created and if the data for a particular row in the SCM column is Fly Ash then

the values of SCM_Slag and SCM_None column will be 0 and value of SCM_Fly Ash column will be 1. So out of the n columns created (here it is three), only one column will have value = 1 and the rest all will have value 0. Thus, all the rows of the SCM feature columns are one hot encoded as shown in
Figure 17. Notice that three columns were added to represent Supplementary Cementitious Material column and the same method is used on all the remaining categorical feature columns.

	SCM_Fly Ash	SCM_None	SCM_Slag
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	0	1	0
...	...	...	...
105	1	0	0
106	1	0	0
107	1	0	0
108	1	0	0
109	1	0	0

110 rows × 3 columns

Figure 17 One hot encoded SCM feature column

After all the categorical features were one hot encoded into numerical values and the actual feature columns of these categorical variables were replaced by encoded columns, there were 17 independent feature columns from which the dependant variables were predicted.

3.4.6 Feature Scaling

Feature scaling is a very important step in any ML pipeline. When different features in the dataset are on a different dimensional range, many of the ML algorithms will not work properly. This is due to the fact that ML algorithms work by computing the distance between different samples in the feature space. If the distribution of one feature is on a very high scale compared to other, the distance calculation will be highly influenced by the feature with a high range. As an example, a summary of the five features in the dataset is given in Table 1.

Table 1 Summary of the data before feature scaling

	Air Entraining Admixture (AEA) (mL)	Air Content (%)	Slump (in)	Water-to-Cement Ratio (WCM)	Water Reducing Admixture (WRA) (mL)
count	110.000000	110.000000	110.000000	110.000000	110.000000
mean	16.431818	5.240909	2.413636	0.387636	31.336364
std	4.839856	0.674475	0.561000	0.017916	37.958277
min	8.000000	3.400000	1.000000	0.330000	0.000000
25%	12.000000	4.725000	2.000000	0.380000	0.000000
50%	16.000000	5.300000	2.500000	0.400000	10.000000
75%	20.000000	5.775000	3.000000	0.400000	60.000000
max	31.000000	6.600000	3.000000	0.400000	125.000000

Table 2 Summary of the data after feature scaling

	Air Entraining Admixture (AEA) (mL)	Air Content (%)	Slump (in)	Water-to-Cement Ratio (WCM)	Water Reducing Admixture (WRA) (mL)
count	1.100000e+02	1.100000e+02	1.100000e+02	1.100000e+02	1.100000e+02
mean	-2.926952e-16	-4.536775e-16	2.301190e-16	-3.874678e-15	1.635056e-16
std	1.004577e+00	1.004577e+00	1.004577e+00	1.004577e+00	1.004577e+00
min	-1.750136e+00	-2.741887e+00	-2.531385e+00	-3.231835e+00	-8.293259e-01
25%	-9.198830e-01	-7.684054e-01	-7.406947e-01	-4.281926e-01	-8.293259e-01
50%	-8.962962e-02	8.801119e-02	1.546505e-01	6.932643e-01	-5.646730e-01
75%	7.406237e-01	7.954857e-01	1.049996e+00	6.932643e-01	7.585914e-01
max	3.023820e+00	2.024257e+00	1.049996e+00	6.932643e-01	2.478835e+00

The features Air Entraining Admixture, Water reducing Admixture have mean of 16.43, 31.33 and standard deviations of 4.83, 37.95 respectively whereas the other three features Air Consent, Slump and Water to Cement Ratio have much smaller means and standard deviations. Hence,

while computing the distance between two features in the feature space, the differences in features Air Consent, Slump and Water to Cement Ratio are negligibly small compared to the difference using Air Entraining Admixture or Water reducing Admixture. So, the input features have to be normalized before applying ML algorithms.

There are many normalization techniques existing to pre-process the data. Standard scalar is one of the widely used methods to normalize the data where features are standardized by removing the mean and scaling them to unit standard deviation.

$$z = \frac{x_i - x_m}{\sigma} \qquad (3.2)$$

where z is the standardized feature, x_i is each data point in feature sample, x_m is the mean of the feature sample, σ is the standard deviation of the original feature sample.

Standard Scaler module in Scikit-learn, a machine learning framework for Python programming language is used to normalize the data in this study [22]. Summary of the five features in the dataset after standard scaling is shown in Table 2. The pre-processed features now have zero mean and unit standard deviation. Generally, feature scaling is done in the pre-processing stage of the pipeline but, it is very common to do it after data analysis since the features would be modified once feature scaling is applied on the data and data analysis might not provide accurate insights. Now the data is ready to be fed into the ML algorithms.

3.5 Evaluation Metrics

This section describes the different techniques and evaluation metrics used in order to compare the ML algorithms applied on the datasets.

3.5.1 Cross Validation

In order to measure the projecting capabilities of any ML model, it should be tested on the data which was not used in training. When the model is trained on the training set and tested on the testing set, the model's performance depends on the kind and variety of data points it sees in the training and testing phases which means the performance changes as the data samples in training and test set changes. So different models cannot be compared to each other as the accuracy of a model is prone to vary if there is a shuffle between data samples of training and testing sets. To avoid this miscalculation, a method called cross validation was used [23].

Cross validation is used to partition the data into training and testing sets. A variety of cross validation methods exist, but the most followed one is the concept of leaving out random data points while training and using the left-out points for testing. This study utilized k-fold cross validation technique to assess model performance. K-fold cross validation consists of randomly grouping the data into k subsets which are also known as folds and k is of our choice. Out of these k subsets, k-1 sets are used to train the model which are training sets and predictions are made on the remaining fold i.e., testing set. The process of training using k-1 folds, testing on the remaining testing fold, and assessing model performance was repeated until every one of the k folds in the

data set has been used for testing. The entire process of splitting data, training and testing until every fold has been used for testing is a single k-fold validation and we obtain as many test results as the number of folds we consider. Each and every data point in the sample is used for both training, testing and different models can be compared to each other to pick the best model for a particular dataset. So, the model performance on the test fold can now be assessed by the Nash and Sutcliffe efficiency values and the root mean squared error of the predicted versus true data.

Nash and Sutcliffe efficiency (NS) is a number that indicates how well a regression model predicts the value of a dependent variable than the mean of that variable. It also shows how much of the variance in the dependent variable is explained by the independent variable and its value ranges from zero to one. It has a value of one when all the values of predicted variables are exactly equal to the actual values and its value decreases as the error between them increase. NS score can also be negative, and it should be noted that NS is the proportion of variance explained by the fit and is not the square of anything. It is not the only metric we should consider assessing the model's performance and more metrics should also be taken into account. The mathematical equation of NS is shown below.

$$NS = 1 - \left[\frac{RSS}{TSS}\right] \qquad (3.3)$$

where RSS is residual sum of squares = $\sum(y_i - \overline{y}_i)^2$, y_i – actual value of dependant variable, $\overline{y}_i$– predicted value of dependant variable, TSS is total sum of squares = $\sum(y_i - y_m)^2$, y_i – actual value of dependant variable, and y_m – mean value of dependant variable.

The most used evaluation metric of all is the Root Mean Squared Error. It tells how much the predicted value can deviate from the actual value of a particular data point on either side i.e., how much less or more than the true value. It is the standard deviation of the errors and is a measure of how spread out these errors are. It is calculated by taking the square root of the average of all the squared residuals or errors. The formula for RMSE is as follows.

$$RMSE = \sqrt{\left[\sum_{i=0}^{n} \frac{(\overline{y}_i - y_i)^2}{N}\right]} \qquad (3.4)$$

where y_i is the actual value of dependant variable, $\overline{y}_i$is the predicted value of dependant variable, and N is the total number of data points.

3.5.2 Grid Search Cross Validation (Grid Search CV)

Each ML algorithm has its own mathematical model and works on its own, background rules and theory. They can be configured by controlling some factors called hyperparameters. We cannot know the best value for a model hyperparameter on a given problem just by looking at the data. We can use rules of thumb, copy values used on other problems or search for the best value by trial-and-error method i.e., by testing them on unseen data. The last method was followed in this project to find out the best parameters for all the ML algorithms using a method or technique called Grid Search Cross Validation.

Grid Search CV is an API available in Scikit-Learn toolkit in Python used to tune hyperparameters of an ML model in order to discover the parameters of that model that result in the most skillful predictions. It takes a) The ML model we want to fine tune the parameters for, b) Different values of all the parameters we want to check on the data, and c) Number of sets we want to divide the data to train and test the model on as inputs i.e., number of k-folds. Grid Search CV builds the given ML model for each of the different hyperparameters, train and test the model on the specified number of k-folds and gives the accuracy results for all those k number of folds having different hyperparameters. The metrics of evaluation can be set as required such as mean squared error, root mean squared error, NS, etc in finding the best hyperparameters. The evaluation metrics was set to root mean squared error in this study for Grid Search CV.

3.5.3 Over Fitting and Under fitting

While finding the best hyperparameters for any ML model, we should be aware of two scenarios that the model might encounter called "over-fitting" and "under-fitting". Overfitting or variance is the case where the overall error is very small but the model being not able to generalize (not producing accurate results) well on the new data and making unreliable predictions. This is due to the model learning too much from the training data set. Solutions to avoid overfitting is not to tune hyperparameters too much to fit the training data well, early stopping while training, etc.

Underfitting or bias is the situation where the model has not learned enough from the training data resulting in a low generalization of the model even on the training data points. Underfitting is as bad as overfitting for generalization of the model. Increasing the training time, tuning the hyperparameters, creating complex and dense algorithms, etc are some of the measures to avoid under fitting [24]. So, to avoid underfitting on the datasets, the hyperparameters of all the models used were tuned to give the best results but extremely high values for hyperparameters were not used so that the models do not overfit the data points.

3.6 Individual Models

This section describes all the individual ML algorithms and the procedure of applying them on the dataset with 28 days Modulus of Rupture as the output variable.

3.6.1 Multi-Variate Linear Regression (MVLR)

Regression analysis is one of the most widely used techniques for predicting continuous output and is the baseline of all the ML algorithms. The generic expression of a multi variate Linear Regression is shown below and multi means that the dependent variable depends on multiple input variables [25].

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \ldots\ldots + b_nx_n \qquad (3.5)$$

where y is the dependent variable which is 28 days Modulus of Rupture here, $x_1, x_2, \ldots, x_n$ are the independent variables in the dataset. b_0 is called the intercept and $b_1, b_2, \ldots, b_n$ are called coefficients or parameters. At first, the values of coefficients were initialized with a real number between 0 and 1 and the dependant variable i.e., 28 days Modulus of Rupture was calculated

substituting the independent variables of a data point. This was done for each and every row i.e., every single sample data point from the dataset and the predicted 28 days Modulus of Rupture values were obtained. The difference between the actual and estimated 28 days Modulus of Rupture values obtained from our regression model are called errors or residuals. The objective is to obtain values of the regression coefficients that would make the sum of the residuals as small as possible, ideally zero. To achieve this, the mean of these squared errors called the cost function was calculated, computed the differentiation and equated it to zero and the best coefficients were found out. The number of iterations this is performed is up to us and the error becomes close to zero as the number increases. At the end the best fit geometric figure was fitted in the feature sample space using these parameters and the dependant variable can be predicted for a new mix sample using this geometric figure.

$$J = \frac{\sum_{i=0}^{n}(y_i - \overline{y}_i)^2}{2M} \qquad (3.6)$$

where J is called cost function or overall error which we try to minimize, y_i – actual value of dependant variable i.e., 28 days Modulus of Rupture, $\overline{y}_i$– predicted value of dependant variable i.e., 28 days Modulus of Rupture, and M is the total number of mix samples in the dataset.

The Linear Regression model has a hyperparameter "normalize" and it should be set to either True or False while building the model. The regression algorithm was obtained from the Scikit-Learn toolkit in Python and Grid search CV was used to find the results of MVLR on the dataset by varying this parameter and setting the number of folds to 10 for predicting 28 days Modulus of Rupture. Since the dataset has 110 points, dividing it into 10 sets gives 11 data points per each set and 9 sets (k-1) were used for training and the reaming set for testing. This process was repeated until all the sets were used for testing and the Gris Search CV results gave better results when normalize was set to True. This is shown in the Table 3 and both gave almost the same results but notice the rank when parameter is False. The columns T1, T2 through T10 represent the test score of each fold from 1 to 10 when Normalize is True and False. Thus, the best hyper parameter for MVLR was found for this dataset.

Table 3 MVLR hyper parameters

Params	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Mean	Rank
True	0.15	0.09	0.60	0.40	0.07	0.29	0.11	0.05	0.05	0.19	0.10	2
False	0.15	0.10	0.60	0.40	0.07	0.30	0.11	0.05	0.05	0.19	0.10	1

3.6.2 Support Vector Regressor (SVR)

There can be a non-linear relationship between input and output features in many real-world datasets which is the case here and the Linear Regression algorithm might not be able to catch these non-linearities. The Support Vector Machine algorithm is typically used to construct the input-output model mapping because it effectively solves nonlinear regression problems and there are many successful use cases of SVM in the field of civil engineering [26]. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that distinctly

classifies the data points where N is the number of features in that sample space. The goal is to find a plane that has the maximum margin i.e., which has the maximum distance between data points of those two classes.

Coming to the regression model of Support Vector Machine, the objective is to minimize the coefficients i.e., the regression coefficients used to find the 28 days Modulus of Rupture as seen in MVLR as opposing to minimize the squared error. SVR gives the flexibility to define how much error i.e., the difference between the actual and estimated 28 days Modulus of Rupture values is acceptable in the model and will find an appropriate hyperplane to fit the mix samples from the dataset. The error term is taken care of in the constraints where we set the absolute error less than or equal to a specified range called the maximum error (ε). The objective function and constraints of SVR are shown below.

$$MIN\ \frac{\beta^2}{2} + C\sum_{i=0}^{n}\varepsilon \qquad (3.7)$$

$$(y_i - b_0 x_i) \leq\ \varepsilon +\ \delta \qquad (3.8)$$

where β is the coefficients vector, C is Regularization parameter, ε is maximum error or margin, δ is allowed deviation from margin, y_i is the actual value of dependant variable i.e., 28 days Modulus of Rupture, x_i is an independent variable for a mix sample in the dataset, and b_0 is regression coefficient for that independent variable.

Grid Search CV was applied on the dataset for SVM which is available in the scikit learn library in python by varying the two most important hyperparameters "Kernel" and "regularization". Kernel functions are used to map the original data points into a higher dimensional space so that the data can be easily separated into different classes. It was varied among three functions i.e., linear, rbf and polynomial. The regularization parameter defines how much error is allowed in predicting 28 days Modulus of Rupture and it was varied from 2 to 100 leaving alternate numbers in between. So, there was a total of 150 combinations obtained by varying these two hyperparameters and out of all Grid search CV showed that the linear kernel with regularization parameter of 88 as the best hyperparameters set suited for this dataset for predicting 28 days Modulus of Rupture.

3.6.3 Decision Tree Regressor (DTR)

The decision tree is one of the most powerful and widely used supervised machine learning algorithms. As the name suggests it uses a tree-like structured model for making decisions by breaking down the data and making a decision based on asking a sequence of questions inferred from the input features in the training set. Decision trees are built by partitioning recursively starting from the root node which is also called a parent node. It is then split into two nodes as left and right child nodes and these can be further split. This process continues till the nodes cannot be split further, and these nodes are called pure nodes. Consider an example of a node checking the value of the Air content feature in the dataset. If it is greater than 4% it might move to the right branch of the tree and if it is less than 4% it might move to the opposite branch of the tree. The

important thing is to know how the nodes are divided and what the optimal splitting point at each node is. This is done based on a concept called "Information gain" and the data is split on the features that have the highest information gain [27].

The formula for information gain is given below and the objective is to split the nodes where the value of this function is maximum. Information gain is the difference between the impurity of the parent node and the sum of impurities of the child nodes. This is for one node divided into two child nodes and if there are multiple nodes in the decision tree, the impurities of all the nodes should be added together.

$$IG_f = I_P - \left(\frac{N_l}{N_p} * I_l + \frac{N_r}{N_p} * I_r\right) \qquad (3.9)$$

where IG_f is the Information gain of the feature at point to perform split, I_P is the impurity measure at parent node, I_l and I_r are the impurity measures of left and right nodes, N_p is the total number of samples at the parent node, and N_l and N_r are the total number of samples in the child nodes.

The most important and effective hyperparameter in the Decision Tree Regressor method is the criteria parameter and it can either be set to mean absolute error (MAE) or mean squared error (MSE). In addition to this, there are some other parameters such as maximum depth of the trees, minimum number of samples per leaf node, minimum samples required to split as node, maximum features to consider, etc. All these parameters are numbers and were not tweaked here as there was no significant impact on the performance of the model and all these parameters were set to default while building the DTR model. It should be noted that decision trees work by always memorizing the training data and sometimes do not perform well on unseen data resulting in overfitting of the model on the dataset. Grid Search CV was applied on our dataset for the DTR algorithm by varying the criteria hyper-parameter between MAE and MSE. There were only two combinations obtained and the results of Grid search CV showed that the DTR performed better to predict 28 days Modulus of Rupture when criteria were set to MAE.

3.7 Ensemble Methods

Ensemble method is an advanced technique in ML where multiple individual algorithms are trained to find a solution to the same problem and all the models are combined to get better results than the individual models [28]. The basic principle is that when weak models are combined, it gives better models and more accurate models. The way these models are combined is two types which are Bagging and Boosting. These two methods are discussed with examples in the below sections by applying these on our dataset.

3.7.1 Random Forest Regressor (RFR)

Bagging also known as bootstrap aggregating is an ensembling process in which the weak ML models are trained independently in parallel and then they are combined to produce the final result. If the prediction is for a continuous variable i.e., regression, then the output of the ensemble model is the average of all the individual models and if it's a classification problem then the class that

receives the majority of the votes i.e., outputted most times will be the final result of the ensemble model.

Random Forest algorithm works on the bagging technique and is also a tree-based algorithm that uses multiple Decision Trees for making decisions [27]. The term "Random" comes from the fact that this algorithm is a forest of randomly created Decision Trees in which many decision trees are constructed and trained in parallel and are combined to produce the end result. As already said that the decision tree has the disadvantage of overfitting and this can be avoided by using multiple trees together to make predictions. RFR averages the results of all the decision trees so that the poor performance of any decision trees can be compensated if there are any trees not performing well to predict 28 days Modulus of Rupture and the overall accuracy of the prediction increases. In addition to this, Random Forest algorithm is very fast and robust than other regression models.

$$S_L = \frac{\sum_{i=0}^{n} w_i}{L} \qquad (3.10)$$

where S_L is the final result of ensemble model, w_i is the output of each individual model, and L is the total number of individual models.

Random Forest also has the same hyperparameters as the Decision Tree model with one extra hyperparameter being the number of trees in the forest. It is the number of trees we want to construct and average the prediction results of 28 days Modulus of Rupture from each tree. The hyperparameters tuned in Random forest regressor are the criteria parameter which should either be MAE or MSE and the number of trees which is a numerical value. Grid Search CV was applied on the dataset using RFR algorithm by varying the criteria hyperparameter between MAE and MSE and number of tress (n estimators) as 10, 20,30 till 200. The results of Grid search CV after running the code showed that the RFR performed better in predicting 28 days Modulus of Rupture when 20 trees were used with criteria set to MSE on this dataset.

3.7.2 Gradient Boosting Regressor (GBR)

Boosting is an Ensembling technique in which individual ML models are trained sequentially to solve a problem. In boosting all the weak learners are fitted on after another on the data and each algorithm depends on the one trained before with each model is fitted giving more importance to data points in the feature space that were wrongly classified by the previous models in the sequence.

Gradient Boosting algorithm works by combining basic models using boosting technique where each model focussing on the difference between the prediction and the ground truth of a dependant variable. As the name suggests, Gradient Boosting algorithm solves the problem by using a gradient descent approach with each weak learner being constructed reduces the error of the prediction based on the learning rate set up [29]. The Decision tree is used as base models in Gradient Boosting. The expression for the output of the ensemble GBR model using a simple squared error as loss function is as follows.

$$F_L = F^1 + \rho * \sum_{i=1}^{n} F(X_i) - \frac{\partial L}{\partial F(X_i)} \qquad (3.11)$$

where $L = (y_i - \overline{y}_i)^2$ i.e, the loss function, F_L is the output of ensemble model, ρ is learning rate, $F(X_i)$ is output of each individual model, y_i – actual value of dependant variable i.e., 28 days Modulus of Rupture, and $\overline{y}_i$– predicted value of dependant variable i.e., 28 days Modulus of Rupture.

Since the weak learners in GBR are the Decision trees, it has the same hyperparameters as them with some extra hyperparameters being the number of trees used in the sequence to build the ensemble model, kind of loss function used and the learning rate. Grid Search CV applied on our dataset using GBR algorithm by varying the number of trees (n estimators) as 10, 30,50 till 200, loss between "least squares" and "Huber" and the learning rate from 0.1, 0.2, 0.3 through 1 gave best results when 30 trees were used with loss function set to least squares and learning rate to 0.1 on this dataset to predict 28 days Modulus of Rupture.

4 Results and discussion

After the hyperparameter tuning of all the ML models applied on the Dtataset_A to predict 28 days Modulus of Rupture determined by RMSE, they were compared with each other to find the best performing ML algorithm on the data. The hyperparameters best suited on this dataset for all the applied algorithms are shown in Table 4 along with the RMSE value obtained for that hyperparameter set. The Modulus of Rupture data in this dataset was measured in PSI so, the RMSE value is the amount of error in PSI the models are showing.

Table 4 Tuned hyperparameters

Algorithm	Best hyper parameters	RMSE (PSI)
MVLR	'normalize' : 'True'	47.85
Support Vector Regressor	'kernel' : 'linear', 'C' : 88	45.76
Decision Tree Regressor	'criterion' : 'mae'	73.09
Random Forest Regressor	'n_estimators' : 20, 'criterion' : 'mse'	56.36
Gradient Boosting Regressor	'n_estimators' : 30, 'loss' : 'ls', 'learning rate' : 0.1	54.32

To compare models with each other the average of the results of each 10-fold cross validation using the best hyperparameters for each algorithm with the evaluation metric as RMSE is taken. We should not choose the best model by simply looking at this single k-fold cross validation results because the results might vary on shuffling the data between folds. So, instead 10-fold cross validation was repeated 100 times with different folds containing data points chosen randomly for each k-fold cross-validation to ensure that the results were not affected by the characteristics of any k-fold split. Each k-fold cross validation gives 10 RMSE and 10 NS values and they were averaged to find one set of k-fold results and similarly, hundred 10-fold cross validation sets gave 100 sets of k-fold results. The obtained 100 different RMSE and NS values of different models can now be compared to find the best algorithm for reliable predictions.

Histograms of RMSE and NS were created for each model for the results of hundred cross validation cycles. Histograms graphically show how many values fall into bins of different ranges and indicate where most of the points are concentrated. In addition to these histograms, pair plots

were also created to assess the model's performance visually. Pair plots are generally created by taking the actual values on the x-axis and the predicted values on the y-axis. A 45^0 angle line is drawn from the origin to check how the predicted values vary from the true values. The RMSE values are normalized by dividing them with the standard deviation of 28 days Modulus of Rupture column (72.50 PSI) to make the final RMSE values around 1 so that it becomes easy to draw comparisons among the different models. If normalized RMSE from a model is less than one, then the model performance is said to be good since the error is less than the spread of the values from the mean of that dependant variable column. A good ML model has low RMSE and high NS scores. Now, the histograms of RMSE, NS from the results of 100 rounds of 10-fold cross validations and pair plots of the fold that gave best RMSE for each model with its best hyperparameters to predict 28 days Modulus of Rupture is shown below.

4.1 MVLR

The results of 10-fold cross validation for multi variate Linear Regression is shown in Figure 18. The mean normalized RMSE is 0.64 which is less than 1 with mean NS of 0.44 and median NS of 0.45. These are the averages of the results of hundred 10-fold cross validation cycles. These values suggest that MVLR is performing reasonably well on our dataset and shows good predictive abilities on the dependant variable but not strong enough for reliable future predictions. The pair plot drawn for the best RMSE fold does not show a good relationship between the predicted and true values and can be seen there are several data points predicted poorly in all the tests and lie far away from the 45^0-angle line. This suggests that the nonlinearities among the features of these mixes in the dataset could not be captured well by MVLR Algorithm. So, we need to go for a non-linear regression algorithm.

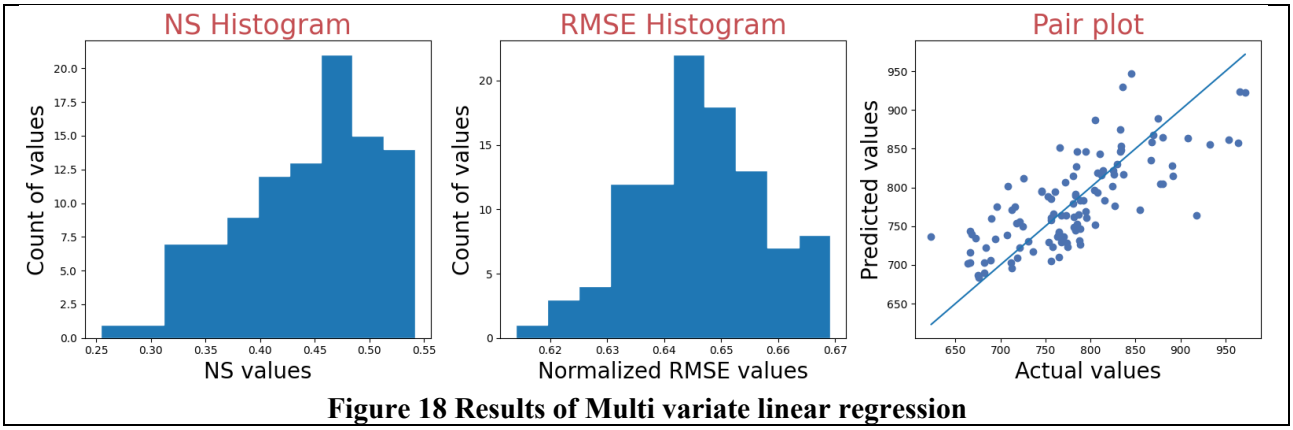

Figure 18 Results of Multi variate linear regression

4.2 Support Vector Machine

The results of the hundred 10-fold cross validation cycles of the Support vector regressor is shown in Figure 19.The mean normalized RMSE obtained is 0.65, the mean NS is 0.43 and median NS is 0.44. There are few points lying far away from the 45^0-angle line in the pair plot drawn for the best RMSE fold for SVR as well. The mean RMSE, NS scores are almost the same for MVLR and SVR, but fewer points lying away from the 45^0-angle line in the pair plot suggest better performance than MVLR. These results imply that SVR is also performing reasonably well but needs a better model to rely on for future predictions.

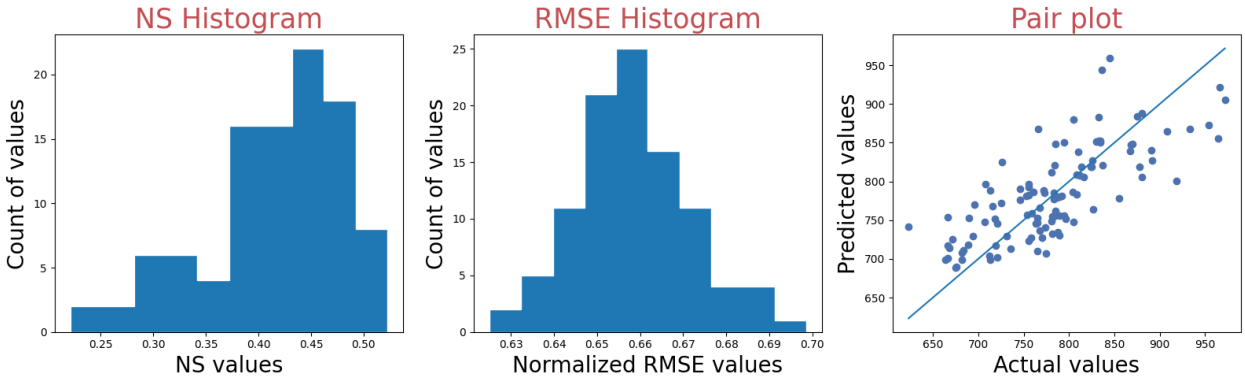

Figure 19 Results of Support Vector Machine

4.3 Decision Tree Algorithm

The histograms of the results of the hundred 10-fold cross validation cycles of Decision Tree regressor is shown in Figure 20. The mean normalized RMSE is 0.91 which is just less than the standard deviation of 28 days Modulus of Rupture column values and the mean and median NS scores is negative. The RMSE values are All the points in the pair plot lie randomly and far away from the central line. These results suggest that DTR is performing really badly on our dataset and shows no predictive abilities at all which means that DTR model cannot be used for future predictions on our data variables. DTR model is showing excellent results when it was trained and tested on the same data points but when different data points were used in train and test sets, it is performing really badly. This scenario is called overfitting as already mentioned and it due to the fact that decision trees work by memorizing the data instead of recognizing actual patterns between variables. Therefore, it sometimes works well with the already seen data and does perform well on the unseen data points.

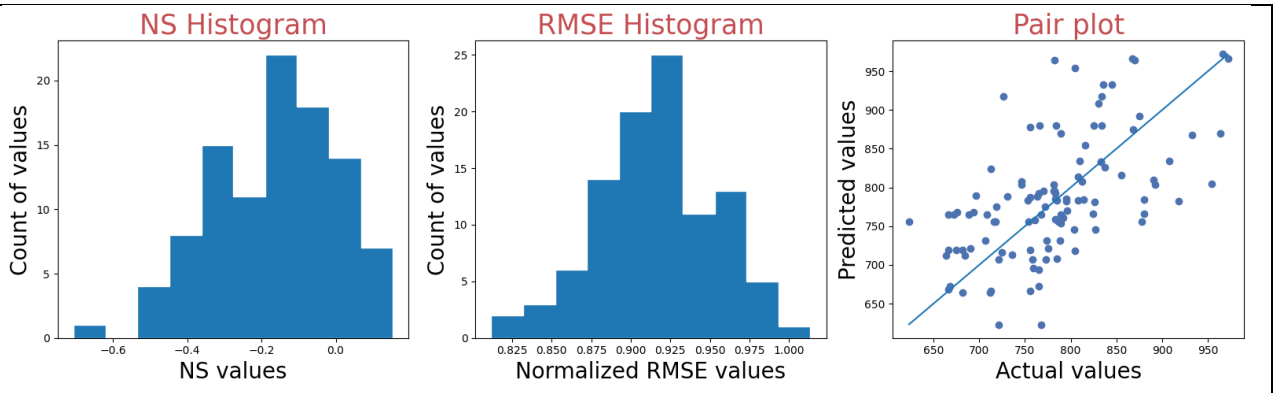

Figure 20 Results of the Decision Tree Model

4.4 Random Forest Algorithm

The mean normalized RMSE from the results of hundred rounds of 10-fold cross validation for Random Forest Regressor is 0.71 which is under 1 and the mean and median NS scores are 0.32 and 0.33, respectively. The histograms obtained are shown in Figure 21. These results are much

better than the Decision tree model which suggests that the over fitting occurring from decision trees is eliminated and the results are comparable to MVLR and SVR models but not surpassing them. From the normalized RMSE histogram, it can be observed that there are several points in the bins of 0.70 – 0.74 RMSE value which is good since RMSE should be low for a good ML model. The overall results are not better than MVLR and SVR models and it should be noted that the base models in RFR are decision trees. Some level of over fitting can be avoided but not all and this was reinforced by the good results obtaining by training and testing on the same data points but poor performance on different train and test sets. Random Forest Algorithm gives the features that are most important to predict the output variables after fitting on a dataset. RFR model determined Air Content, Slump and Water Reducing Admixture as the top three most important features in predicting 28 days Modulus of Rupture.

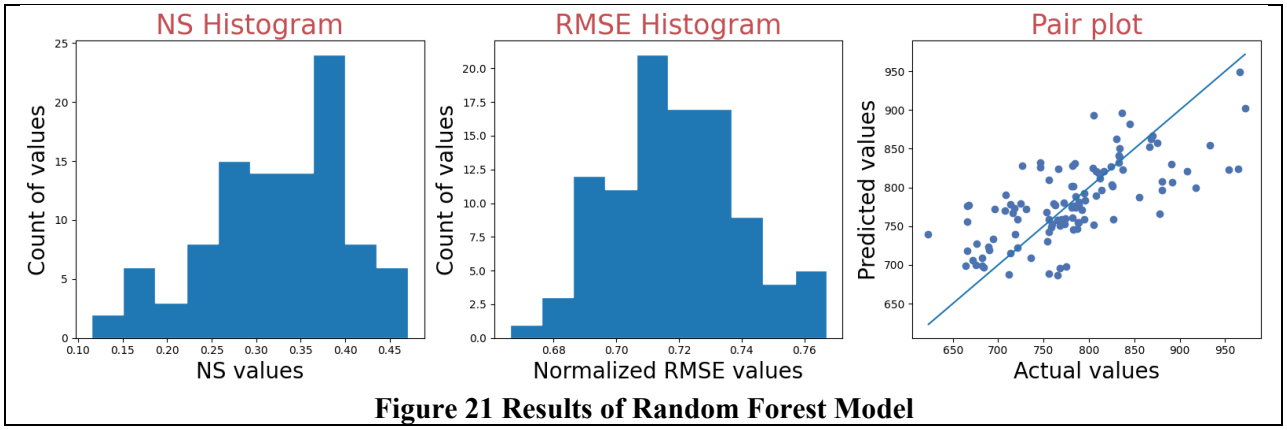

Figure 21 Results of Random Forest Model

4.5 Gradient Boosting Algorithm

The mean normalized RMSE of Gradient Boosting Regressor is 0.71 which is less than 1 with mean NS score of 0.34 and median NS score of 0.35. The histograms and pair plot is shown in Figure 22. These results are much better than Decision tree model and are very similar to the results of the Random Forest Regressor model. The performance of Gradient Boosting model on our dataset did not outshine the MVLR and SVR models since the weak learners in Gradient Boosting Algorithm are decision trees and there would be some overfitting in GBR as well.

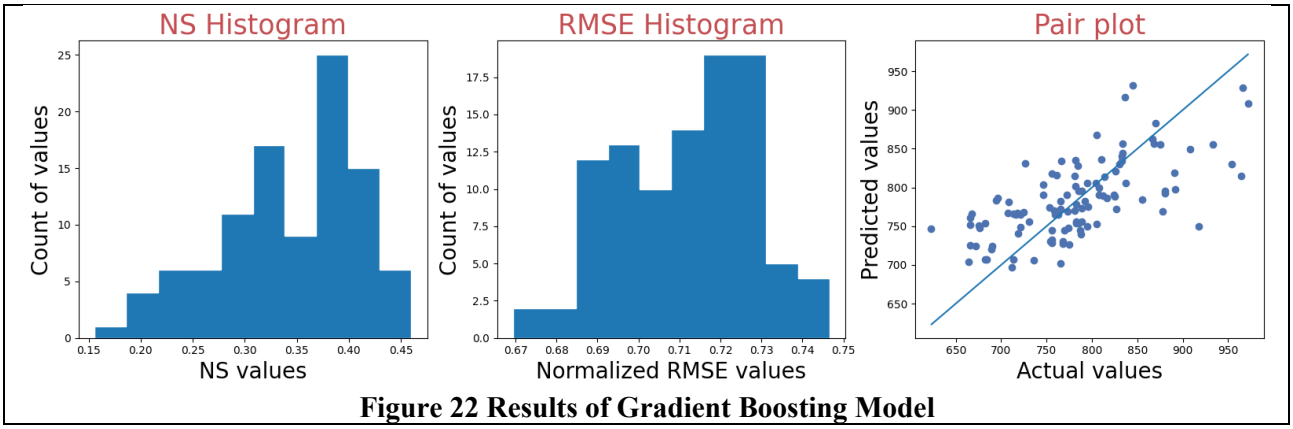

Figure 22 Results of Gradient Boosting Model

Table 5 ML models result for 28 days Modulus of Rupture

Algorithm	Median NS	Mean NS	Normalized RMSE Range	Mean Normalized RSME	Best RMSE (PSI)
MVLR	0.45	0.44	0.61 – 0.66	0.64	44.51
Support Vector Machine	0.44	0.43	0.62 – 0.69	0.65	45.31
Decision Tree	-0.14	-0.16	0.81 – 1.01	0.91	58.86
Random Forest	0.33	0.32	0.66 – 0.77	0.71	48.29
Gradient Boosting	0.35	0.34	0.66 – 0.74	0.71	48.53

The results of all the applied ML models with 28 days Modulus of Rupture as the predicted variable are shown in Table 5 along with the best RMSE score out of the 100 folds. The Support Vector Machine and MVLR have almost the same RMSE and NS scores but the pair plots indicated that Support Vector Machine made be tter predictions than all the models fitted on the dataset and MVLR seems to be the second-best model. The normalized RMSE (i.e., divided by standard deviation) of all the models is less than one which suggests a good performance since the predictions are better than the spread (i.e., standard deviation) of the variable values. MVLR and SVR models have the best RMSE and NS scores. They are 30% more accurate than the other models used on the dataset and this suggests that they are able to predict future data points but the Ns score needs to improve for more accurate predictions. It can be seen that there are some negative values of NS in the histograms. This meant that the prediction is worse than the mean of actual values in that particular output feature column.

Table 6 ML models results of all properties

Predicted Property	Algorithm	Median NS	Mean NS	Normalized RMSE Range	Mean Normalized RSME
28 days Compression Strength	MVLR	0.41	0.40	0.64 – 0.72	0.67
	Support Vector Machine	0.43	0.42	0.63 – 0.69	0.67
	Decision Tree	-0.31	-0.31	0.85 – 1.12	0.98
	Random Forest	0.33	0.32	0.68 – 0.78	0.73
	Gradient Boosting	0.23	0.22	0.71 – 0.81	0.77
28 days Modulus of Elasticity	MVLR	0.06	0.05	0.85 – 0.93	0.88
	Support Vector Machine	-0.07	-0.08	0.93 – 0.99	0.97
	Decision Tree	-1.06	-1.10	1.07 – 1.51	1.29
	Random Forest	-0.01	-0.04	0.88 – 0.99	0.94
	Gradient Boosting	-0.03	-0.03	0.89 – 0.98	0.94
28 days Poisson's Ratio	MVLR	-0.06	-0.09	0.89 – 0.98	0.94
	Support Vector Machine	-0.10	-0.13	0.95 – 1.01	0.98
	Decision Tree	-1.23	-1.23	1.16 – 1.45	1.29
	Random Forest	-0.14	-0.14	0.90 – 1.03	0.96
	Gradient Boosting	-0.10	-0.12	0.88 – 1.03	0.94
28 days Splitting Tensile Strength	MVLR	0.02	-0.04	0.81 – 0.89	0.85
	Support Vector Machine	0.07	0.10	0.77 – 0.87	0.82
	Decision Tree	-1.05	-1.10	0.97 – 1.40	1.18
	Random Forest	0.02	0.01	0.82 – 0.94	0.87
	Gradient Boosting	-0.12	-0.14	0.87 – 1.00	0.93
28 days Coefficient of Thermal Expansion	MVLR	-0.09	-0.11	0.88 – 0.99	0.94
	Support Vector Machine	-0.08	-0.10	0.89 – 0.99	0.94
	Decision Tree	-0.10	-0.13	0.92 – 0.99	0.96
	Random Forest	-0.09	-0.10	0.87 – 0.97	0.93
	Gradient Boosting	-0.10	-0.13	0.92 – 0.99	0.96

The similar process was followed by taking Dataset_B with all the ML models applied to predict its output variables as well as the 28 days CTE in Dataset_A. Table 6 summarizes the results of all the applied ML models on the remaining five properties measured after 28 days. Except for predicting 28 days Modulus of Elasticity and 28 days Poisson's ratio Support Vector Machine algorithm performed better than all the other models in all the cases as it has the highest NS score and least RMSE value out of all. MVLR came out as the best model for predicting the Modulus of Elasticity and Poisson's ratio. The results of all the predicted properties at 7^{th}, 14^{th}, 90^{th} day showed similar behaviour with SVR and MVLR coming to be the best models on these datasets. These results suggest that they can be used to predict new data at least for similar systems by improving NS score. A previous study conducted on the same dataset stated that Random Forest algorithm

with 90 trees as an effective model [3]. This study determines Support Vector Machine as the best-suited model on this dataset which has very less computational complexity than the Random Forest model.

Decision Tree, Random Forest and Gradient Boosting algorithms showed better performance when trained and tested on the same data for all the properties with Random Forest coming out as the best model in this case out of the five algorithms used. As the data available for ML algorithms increases their performance would improve. Though Random Forest and Gradient Boosting are more powerful non-linear algorithms, their performance was not up to the par on these datasets because they were overfitting the training data due to the fact that there are just 110 data points in the datasets and their testing accuracy increases as more data points are available for training consequently reducing overfitting. The 110 data points available in our dataset were a result of several experiments and the widely applied nonlinear Machine Learning Algorithms can become useful tools for accurate prediction of different properties of concrete and can reduce the number of such physical experiments.

5 Conclusions and Future work

Various Machine Learning Algorithms were used to study their behavior and performance on different properties of concrete. The machine learning models used in this study were effective in predicting six mechanical properties of concrete for different mixes which are compressive strength, splitting tensile strength, modulus of rupture, modulus of elasticity, coefficient of thermal expansion, and Poisson's ratio. This study started with collecting the data points from the experiments conducted to measure properties of concrete in the Mechanistic Empirical Pavement Design Guide [9]. The input and output variables in the datasets were examined first to find the relations between them. The process of applied ML models was clearly explained for 28 days Modulus of Rupture and the results of all the other properties were presented. Analysis of NS score and RMSE of all the predictions after applying ML models demonstrated that Support Vector Machine model was the best performing one in most of the cases. The predictions of SVM were more accurate than the Multi Variate Linear Regression, Decision Tree, Random Forest and Gradient Boosting Algorithms used here. This shows the need for a powerful non-linear algorithm to find patterns in non-linear data and to predict new concrete mixes.

The NS scores and normalized RMSE values suggest that machine learning can be effectively applied for predicting a wide range of concrete properties Therefore, ML models trained on a lot of experimental data available in different databases can be used to predict future data points. The next step is to collect more data from the experiments being conducted and use advanced ML algorithms and Artificial Neural Networks (ANN) as well which is a powerful tool for solving very complex problems and are expected to outperform the Machine Learning models as they have a high learning curve, but they require thousands of data points. With further improved Machine Learning Algorithms, computational power and more experimental data, the predictions made could reach the accuracy of experimental measurements. Thus, reducing the number of experiments to be performed.

6 Bibliography

[1] E. Vintzileou and E. Panagiotidou, “An empirical model for predicting the mechanical properties of FRP-confined concrete,” *Construction and Building Materials*, vol. 22, no. 5, pp. 841–854, May 2008, doi: 10.1016/j.conbuildmat.2006.12.009.

[2] M. Jalal, “Soft computing techniques for compressive strength prediction of concrete cylinders strengthened by CFRP composites,” *Science and Engineering of Composite Materials*, vol. 22, no. 1, pp. 97–112, Jan. 2015, doi: 10.1515/secm-2013-0240.

[3] V. Nilsen, L. T. Pham, M. Hibbard, A. Klager, S. M. Cramer, and D. Morgan, “Prediction of concrete coefficient of thermal expansion and other properties using machine learning,” *Construction and Building Materials*, vol. 220, pp. 587–595, Sep. 2019, doi: 10.1016/j.conbuildmat.2019.05.006.

[4] K. Smarsly, K. Dragos, and J. Wiggenbrock, “Machine learning techniques for structural health monitoring,” Jul. 2016. Accessed: May 2020. [Online]. Available: https://www.ndt.net/events/EWSHM2016/app/content/Paper/21_Smarsly_Rev1.pdf.

[5] A. Mosavi, T. Rabczuk, and A. R. Varkonyi-Koczy, “Reviewing the Novel Machine Learning Tools for Materials Design,” *Advances in Intelligent Systems and Computing*, pp. 50–58, Sep. 2017, doi: 10.1007/978-3-319-67459-9_7.

[6] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, “Machine learning in materials informatics: recent applications and prospects,” *npj Computational Materials*, vol. 3, no. 1, Dec. 2017, doi: 10.1038/s41524-017-0056-5.

[7] C. Meyer, “Concrete Materials and Sustainable Development in the USA,” Structural Engineering International, vol. 14, no. 3, pp. 203–207, Aug. 2004, doi: 10.2749/101686604777963757.

[8] P. Chopra, R. K. Sharma, M. Kumar, and T. Chopra, “Comparison of Machine Learning Techniques for the Prediction of Compressive Strength of Concrete,” *Advances in Civil Engineering*, Apr. 12, 2018. https://www.hindawi.com/journals/ace/2018/5481705/ (accessed May2020).

[9] M.-C. Kang, D.-Y. Yoo, and R. Gupta, “Machine learning-based prediction for compressive and flexural strengths of steel fiber-reinforced concrete,” *Construction and Building Materials*, vol. 266, p. 121117, Jan. 2021, doi: 10.1016/j.conbuildmat.2020.121117.

[10] I.-C. Yeh and L.-C. Lien, “Knowledge discovery of concrete material using Genetic Operation Trees,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 5807–5812, Apr. 2009, doi: 10.1016/j.eswa.2008.07.004.

[11] E. Alpaydin, *Introduction to machine learning*. Cambridge (USA): MIT Press, 2014.

[12] A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 1310-1315.

[13] M. E. Celebi and K. Aydin, Eds., *Unsupervised Learning Algorithms*. Cham: Springer International Publishing, 2016.

[14] J. Effinger, R. Li, J. Silva, and S. Cramer, "Laboratory Study of Concrete Properties to Support Implementation of the New AASHTO Mechanistic-Empirical Pavement Design Guide," 2012. Accessed: May 2020. [Online]. Available: https://wisconsindot.gov/documents2/research/WisDOT-WHRP-project-0092-10-11-final-report.pdf

[15] "STATE OF WISCONSIN STANDARD SPECIFICATIONS FOR HIGHWAY AND STRUCTURE CONSTRUCTION 2010 Edition." Accessed: May 2020. [Online]. Available: https://wisconsindot.gov/Pages/doing-bus/eng-consultants/cnslt-rsrces/rdwy/ss-archive.aspx.

[16] M. Harrison and T. Petrou, *PANDAS 1.X COOKBOOK : practical recipes for scientific computing, time series and ... exploratory data analysis using python.*, 2nd ed. Packt Publishing, 2020.

[17] N. Ranga Suri, N. Murty M and G. Athithan, Outlier detection--techniques and applications.

[18] L.Sunitha , Dr M.BalRaju, Dr J.Sasikiran, E.Venkat Ramana, "Automatic Outlier Identification in Data Mining Using IQR in Real-Time Data", International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 6, June 2014.

[19] K. P. Vatcheva and M. Lee, "Multicollinearity in Regression Analyses Conducted in Epidemiologic Studies", *Epidemiology: Open Access*, vol. 06, no. 02, 2016. Available: 10.4172/2161-1165.1000227.

[20] [4]G. Duncan, W. Berry and S. Feldman, "Multiple Regression in Practice", *Journal of Marketing Research*, vol. 23, no. 3, p. 309, 1986. Available: 10.2307/3151494.

[21] H. Alkharusi, "Categorical Variables in Regression Analysis: A Comparison of Dummy and Effect Coding," *International Journal of Education*, vol. 4, no. 2, p. 202, Jun. 2012, doi: 10.5296/ije.v4i2.1962.

[22] P. Fabian, V. Gaël , G. Alexandre , M. Vincent , T. Bertrand , G. Olivier , B. Mathieu , P. Peter , W. Ron , D. Vincent , V. Jake , P. Alexandre , C. David , B. Matthieu , P. Matthieu and D. Édouard , "Scikit-learn: Machine Learning in Python", *JMLR,* vol. 12, pp. 2825-2830, 2011. Available: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[23] G. Vanwinckelen and H. Blockeel, "On Estimating Model Accuracy with Repeated Cross-

Validation." Accessed: Jun. 2020. [Online]. Available: https://limo.libis.be/primo-explore/fulldisplay?docid=LIRIAS1655861&context=L&vid=Lirias&search_scope=Lirias&tab=default_tab&lang=en_US&fromSitemap=1.

[24] H. Khalaf, J. Dr, Z. Rafiqul, and Khan, "METHODS TO AVOID OVER-FITTING AND UNDER-FITTING IN SUPERVISED MACHINE LEARNING (COMPARATIVE STUDY)." Accessed: Jun. 2020. [Online]. Available: https://www.researchgate.net/publication/295198699_METHODS_TO_AVOID_OVER-FITTING_AND_UNDER-FITTING_IN_SUPERVISED_MACHINE_LEARNING_COMPARATIVE_STUDY.

[25] D. N. Gujarati, *Linear regression : a mathematical introduction*. Thousand Oaks, California: Sage Publications, Inc, 2019, pp. 1–15.

[26] K. Yan and C. Shi, "Prediction of elastic modulus of normal and high strength concrete by support vector machine," *Construction and Building Materials*, vol. 24, no. 8, pp. 1479–1485, Aug. 2010, doi: 10.1016/j.conbuildmat.2010.01.006.

[27] J. T. Vanderplas, *Python data science handbook : essential tools for working with data*. Beijing Etc.: O'reilly, Cop, 2017.

[28] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*, vol. 1857, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15.

[29] Zemel, R. and Pitassi, T., 2001. A Gradient-Based Boosting Algorithm for Regression Problems. Advances in Neural Information Processing Systems 13, 696–702. https://proceedings.neurips.cc/paper/2000/file/8d9fc2308c8f28d2a7d2f6f48801c705-Paper.pdf.

Appendix A: Major Software Packages Used

Python

A python is an object-oriented, high level, interpreted, open-source programming language.
Documentation: https://docs.python.org/3/

Pandas

pandas is an open source, high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
Documentation: http://pandas.pydata.org/pandas-docs/stable/

Numpy

NumPy is the fundamental package for scientific computing with Python.
Documentation: https://www.numpy.org/devdocs/

Sci-kit Learn

Sci-kit learn is an open-source Python library for data preprocessing, data analytics and machine learning.
Documentation: https://scikit-learn.org/stable/documentation.html

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
Documentation: https://matplotlib.org/users/index.html

Appendix B: Code for Data Analysis

```
import pandas as pd  # Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
%matplotlib inline

df.describe() # Measures of Central tendency

def Feature_analysis(x): # Function to find number of ouliers and to draw feature plots

  Q1 = df[x].quantile(q=0.25)
  Q3 = df[x].quantile(q=0.75)
  IQR = stats.iqr(df[x])

  print('1st Quartile of ' + x + ' :' , Q1)
  print('3rd Quartile of ' + x + ' :', Q3)
  print('Inter Quartile range of ' + x + ' :', IQR, '\n')

  Lower_outlier = Q1 - 1.5*(IQR)
  upper_outlier = Q3 + 1.5*(IQR)

  print('Lower outlier value in ' + x + ' :', Lower_outlier)
  print('upper outlier value in ' + x + ' :', upper_outlier, '\n')

  print('Number of lower outliers in ' + x + ' :' ,df[df[x]<Lower_outlier][x].count())
  print('Number of upper outliers in ' + x + ' :' ,df[df[x]>upper_outlier][x].count())

  print('% of lower outliers in ' + x + ' :', round(df[df[x]<Lower_outlier][x].count()*100/len(df)))
  print('% of upper outliers in ' + x + ' :',
round(df[df[x]>upper_outlier][x].count()*100/len(df)),'\n' )

  fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(16,5))

  # Distribution plot
  sns.distplot(df[x], ax=ax1)
  ax1.set_xlabel(x, fontsize=15)
  ax1.set_title('Distribution plot', fontsize=20)
  ax1.tick_params(labelsize=15)

  # Box plot
  sns.boxplot(df[x], ax=ax2, orient='v' )
  ax2.set_ylabel(x, fontsize=15)
  ax2.set_title( x +' ' + 'Box plot', fontsize=20)
```

```
  ax2.tick_params(labelsize=15)

  # Histogram
  ax3.hist(df[x])
  ax3.set_xlabel(x, fontsize = 15)
  ax3.set_title('Histogram', fontsize = 20)
  ax3.tick_params(labelsize=15)

  plt.tight_layout()

rows = 3 # distribution plots
cols = 3
fig, ax  = plt.subplots(nrows=rows, ncols=cols, figsize = (22, 12))

columns = df.columns[6:]
index = 0

for i in range(rows):
  for j in range(cols):
    sns.distplot( df[columns[index]] ,ax = ax[i][j])
    index += 1

plt.figure(figsize = (12,8)) # Correlation Matrix
sns.heatmap(correlation, annot=True)
plt.title("Correlation among Features", fontsize = 20)

fig, ax = plt.subplots(figsize=(15,8)) # Visualizations
sns.set_style(style='darkgrid')
sns.scatterplot(x = df['Air Entraining Admixture (AEA) (mL)'], y= df['Modulus of Rupture (PSI)
(28 Days)'], hue=df['Water Reducing Admixture (WRA) (mL)'], size= df['Air Content (%)'],
sizes=(30, 500), palette='magma')
ax.set_title("Modulus of Rupture (28 days) vs(AEA, WRA, Air Content)")
ax.legend(loc = 'upper left',bbox_to_anchor=(1,1))

df = pd.get_dummies(df) # One hot encodiing

ss = StandardScaler() # Feature Scaling
x.iloc[ : , 0:5]= ss.fit_transform(x.iloc[ : , 0:5])
```

Appendix C: Code for Grid Search CV on all ML models

```
from sklearn.linear_model import LinearRegression # Importing necessary libraries
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV

models = {
   'Linear_Regression': {'model': LinearRegression(), 'parameters' : {'normalize': ['True', 'False']}
} ,

   'Support_Vector_Regressor':  {'model': SVR(), 'parameters': { 'C':list(range(2,101, 2)), 'kernel':
['linear', 'rbf', 'poly'] } }   ,

   'Decision_Tree_Regressor': {'model': DecisionTreeRegressor(), 'parameters' : {'criterion':
['mse', 'mae']}   } ,

   'Random_Forest_Regressor': {'model': RandomForestRegressor(), 'parameters' :
{'n_estimators':list(range(10, 201, 10)), 'criterion': ['mse', 'mae'] }  } ,

   'Gradeint_Boosting' : {'model': GradientBoostingRegressor(), 'parameters':
{'n_estimators':list(range(10, 201, 20)), 'loss':['ls', 'huber'], 'learning_rate':list(np.linspace(0.1, 1,
10,)) }}
}

scores = []

for model_name, value in models.items():

  clf = GridSearchCV( value['model'] , value['parameters'], cv=10, return_train_score=False,
scoring='neg_root_mean_squared_error')
  clf.fit(x,y)

  scores.append(
     { 'model': model_name,
   'best_parameters' : clf.best_params_ ,
   'best_score' : -(clf.best_score_)
  })
model_selection_df = pd.DataFrame(scores)
```

Appendix D: Code for hundred rounds 10-fold Cross Validation

```
from math import sqrt
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

mean_r2_scores = []
mean_rmse_scores = []

for k in range(1,101):


  kf = KFold(n_splits=10, shuffle=True, random_state=k)

  #model = LinearRegression(normalize=False)
  model = SVR(C=96, kernel='linear')
  #model = DecisionTreeRegressor(criterion='mae')
  #model = RandomForestRegressor(n_estimators=130,criterion='mse' )
  #model = GradientBoostingRegressor(n_estimators=130, loss = 'ls', learning_rate=0.3)

  r2_score = cross_val_score(model, x,y, cv=kf, scoring='r2')
  rmse_score = -(cross_val_score(model, x,y, cv=kf, scoring='neg_root_mean_squared_error'))


  print('r2 scores for',k, 'iteration : '  ,r2_score)
  print('rmse scores for', k ,'iteration :',rmse_score)
  print('mean r2 score for',k, 'iteration :' ,np.mean(r2_score))
  print('mean rmse score for',k, 'iteration :',  np.mean(rmse_score), '\n')

  mean_r2_scores.append(np.mean(r2_score))
  mean_rmse_scores.append(np.mean(rmse_score))
```

Appendix E: Code for plotting R2, RMSE histograms and pairplots

```
fig, (ax1, ax2, ax3) = plt.subplots(ncols= 3, nrows= 1, figsize=(16,5))

best_rmse_point = np.argmin(mean_rmse_scores)

ax1.hist(mean_r2_scores)
ax1.set_title('R-squared Hisotgram',fontsize = 25, c='r')
ax1.set_xlabel('R2 values', fontsize = 20 )
ax1.set_ylabel('Count of values',fontsize = 20 )

std = df['Compressive Strength (PSI) (7 days)  '].std()
normalized_mean_rmse_scores = [x/std for x in mean_rmse_scores]
ax2.hist(normalized_mean_rmse_scores)
ax2.set_title('RMSE Hisotgram',fontsize = 25, c='r')
ax2.set_xlabel('Normalized RMSE values', fontsize = 20 )
ax2.set_ylabel('Count of values',fontsize = 20 )

from sklearn.model_selection import KFold
kf = KFold(n_splits=10, shuffle=True, random_state = best_rmse_point)

#a = [np.min(df['CTE (mm/mm/0C) (28 Days)']), np.max(df['CTE (mm/mm/0C) (28 Days)'])]
plt.plot(a, a)

x = np.array(x)
for i,j in kf.split(x):
  x_train = x[i]
  y_train = y[i]
  x_test = x[j]
  y_test = y[j]

  #model = LinearRegression(normalize=False)
  model = SVR(kernel='linear', C= 100)
  #model = DecisionTreeRegressor(criterion='mae' )
  #model = RandomForestRegressor(n_estimators=110, criterion='mae')
  #model = GradientBoostingRegressor(n_estimators=10, loss = 'ls', learning_rate=0.2)
  model.fit(x_train, y_train)
  plt.scatter(  y_test,model.predict(x_test), c='b')

ax3.set_title('Pair plot',fontsize = 25, c='r' )
ax3.set_xlabel('Actual values', fontsize = 20 )
ax3.set_ylabel('Predicted values',fontsize = 20 )

plt.tight_layout()
plt.show()
```