

PREDICTING COMPRESSIVE STRENGTH OF CONCRETE USING IBM WATSON MACHINELEARNING

1.INTRODUCTION

1.1 Overview

Considering that compressive strength (CS) is an important mechanical property parameter in many design codes, in order to ensure structural safety, concrete CS needs to be tested before application. However, conducting CS tests with multiple influencing variables is costly and time consuming. To address this issue, a machine learning-based modelling framework is put forward in this work to evaluate the concrete CS under complex conditions. The influential factors of this process are systematically categorized into five aspects: man, machine, material, method and environment.

The CS of concrete is obtained by testing specifically prepared and cured cubic or cylindrical specimens using a compression test instrument, which is cumbersome, time-consuming and costly in the entire experimental process. To improve this situation, empirical regression methods and numerical simulation have been developed to predict concrete CS based on the design recipe. The concrete production process is affected by several factors, which have strong nonlinear relationships with the product CS and are strongly interrelated. With the rapid development of machine learning, there is a trend to employ data-driven techniques for concrete CS prediction. Compared with the conventional regression methods, machine learning-based approaches adopt suitable algorithms to automatically “learn” from the process data, “distinguish” important influential factors from the interfering factors, approximate the intricate process mechanism with deterministic mathematical forms, and perform prediction with high accuracy over a specified confidence interval. To date, various machine learning algorithms have been applied to studying the correlations between the concrete recipe and the product CS.

Researchers have also proved that besides the type and basic properties of raw materials, environmental factors, such as temperature, and relative humidity, also significantly determine the concrete CS. To exclude the influence from these environmental factors. Additionally, the recipe and environmental variables may not be sufficient in covering all the possible factors influencing concrete CS. It has been proved in many areas that to comprehensively evaluate the quality of an engineering product, the influences from man, material, machine, method and environment (shortened to 4M1E), should be considered. Each of these factors further represents the aggregation of various detailed influential factors.

1.2 Purpose

The combined effect of the above factors (4MIE) poses new challenges for the accurate prediction of concrete CS. Since the influence from the “material” aspect is already inherently complicated, when the joint influences from the other four aspects are included, the problem dimension becomes high, and it will be more difficult to manually identify the most relevant factors from the interfering ones.

Construction of both traditional regression models and machine learning models to predict the 28-day CS of no-slump concrete, based on the concrete ingredients.

Computer-aided feature selection techniques provide an alternative solution to this situation. Common feature selection algorithms can be categorized into two classes based on the search strategies—the filter methods and the wrapper methods. Applied principal component analysis (PCA) to reduce the noise in the input space and consequently improve the model predictive performance.

Principal component analysis is a typical filter method that performs feature selection based on the statistical performance of the original dataset and is independent of the subsequent learning algorithm. Different from the filter methods, the wrapper methods tightly couple the subsequent prediction algorithm with the feature selection process. In other words, the feature selection process is optimized based on the feedback from the subsequent algorithm performance. Commonly used wrapper methods include genetic algorithms (GAs) and particle swarm optimization (PSO).

2. LITERATURE SURVEY

2.1 Existing Problem

In the present study, a regression model was investigated as a performance prediction model for predicting the concrete compressive strength. Moreover, the effects of the changes of the coefficients of regression model of the performance curve were also examined. For this purpose, multiple regression analysis was carried out for predicting the compressive strength of concrete using four variables, namely, water-cementations ratio, fine aggregate cementitious ratio, coarse aggregate-cementitious ratio and cementitious content. Regression models were developed for concrete with medium and high workability at different curing ages (28, 56 and 91days). For models developed for compressive strength prediction at 56 days and 91 days, the compressive strengths at lower ages were also considered as a parameter.

For the experimental data obtained, the data sets on the dependent variable were normally distributed for each of the possible combinations of the level of the X variables, it was deemed suitable to use regression models for prediction of concrete strengths. Bayrak and Akgül predicted the lifetime performance and remaining service

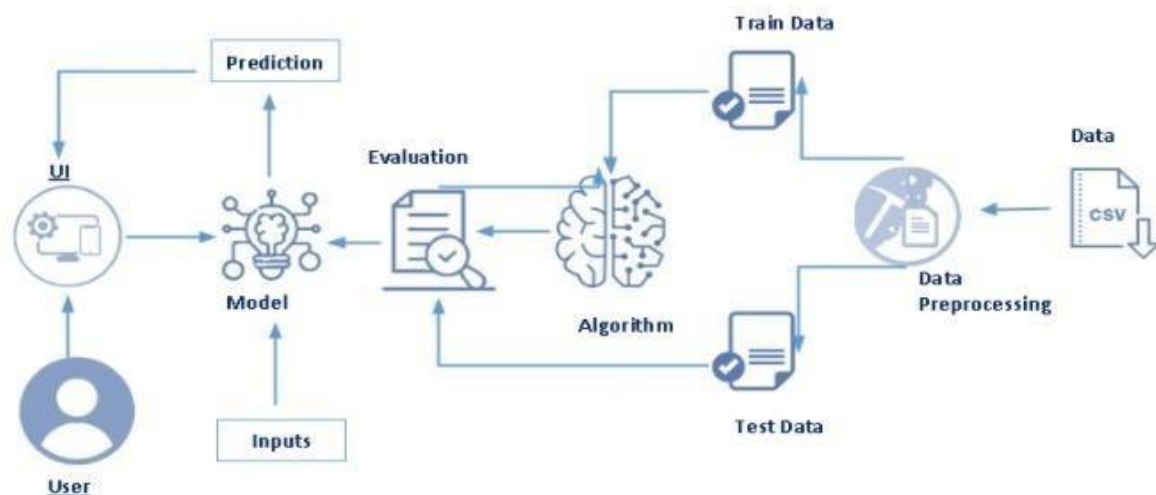
life of a bridge system using regression models, so that the best maintenance and repair strategies which kept the system safe can be obtained. Many attempts had earlier been made to obtain a suitable mathematical model that was capable of predicting the strength of concrete at various ages with good accuracy.

2.2 Proposed Solution

A mathematical model for prediction of compressive strength of concrete will be performed using statistical analysis for the concrete data obtained from experimental work done in this study. The multilinear regression yielded excellent correlation coefficient for prediction of compressive strength at different ages (3, 7, 14, 28, 91 days). The coefficient of correlation was 99.99% for each strength (at each age). Also, the model gives high correlation for strength prediction of concrete with different types of curing. Given the characteristics of the concrete production process, multi linear regression is considered a very suitable modelling algorithm, due to its versatile merits, such as its good tolerance for outliers and noises, its ability to avoid overfitting, and its ability to deal with multicollinearity.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/ Software designing

Software Requirements:

- Anaconda Navigator • Keras
- Flask

Hardware Requirements:

- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 8 GB
- Display : 14.1 "Color Monitor(LCD, CRT or LED)
- Clock Speed : 1.67 GHz

4.EXPERIMENTAL INVESTIGATIONS

DATA PREPROCESSING

Importing Libraries

```
In [1]: import pandas as pd
import numpy as np
from collections import Counter as c
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error, mean_absolute_error
import pickle
from sklearn.linear_model import LinearRegression
```

Loading Dataset

```
In [2]: data=pd.read_csv(r"F:\FLASK\concrete.csv")
```

```
In [3]: data.head()
```

Out[3]:

	Cement (component 1) (kg in a m³ mixture)	Blast Furnace Slag (component 2)(kg in a m³ mixture)	Fly Ash (component 3) (kg in a m³ mixture)	Water (component 4) (kg in a m³ mixture)	Superplasticizer (component 5)(kg in a m³ mixture)	Coarse Aggregate (component 6)(kg in a m³ mixture)	Fine Aggregate (component 7)(kg in a m³ mixture)	Age (day)	Concrete compressive strength(MPa, megapascals)
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30

```
In [4]: data.tail()
```

Out[4]:

	Cement (component 1) (kg in a m³ mixture)	Blast Furnace Slag (component 2)(kg in a m³ mixture)	Fly Ash (component 3) (kg in a m³ mixture)	Water (component 4) (kg in a m³ mixture)	Superplasticizer (component 5)(kg in a m³ mixture)	Coarse Aggregate (component 6) (kg in a m³ mixture)	Fine Aggregate (component 7) (kg in a m³ mixture)	Age (day)	Concrete compressive strength(MPa, megapascals)
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.18
1027	148.5	139.4	108.8	192.7	6.1	892.4	780.0	28	23.70
1028	159.1	166.7	0.0	175.6	11.3	989.8	788.9	28	32.77
1029	280.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40

Renaming columns

```
In [5]: data.columns = [col[:col.find("(")].strip() for col in data.columns]
data.head(1)
```

Out[5]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Concrete compressive strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99

Info of the data

```
In [6]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
Cement                1030 non-null float64
Blast Furnace Slag    1030 non-null float64
Fly Ash               1030 non-null float64
Water                1030 non-null float64
Superplasticizer      1030 non-null float64
Coarse Aggregate      1030 non-null float64
Fine Aggregate        1030 non-null float64
Age                  1030 non-null int64
Concrete compressive strength  1030 non-null float64
dtypes: float64(8), int64(1)
memory usage: 72.5 kB
```

```
In [7]: data.describe()
```

Out[7]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Concrete compressive strength
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.885825	54.188350	181.567282	6.204660	972.918832	773.580485	45.662136	35.817861
std	104.606364	86.278342	63.997004	21.354219	5.973841	77.753954	80.176980	63.169612	16.706742
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.900000	118.300000	192.000000	10.200000	1028.400000	824.000000	56.000000	46.135000
max	540.000000	369.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	366.000000	82.600000

Null Values

```
In [16]: data.isnull().any() #it will return true if any columns is having null values
```

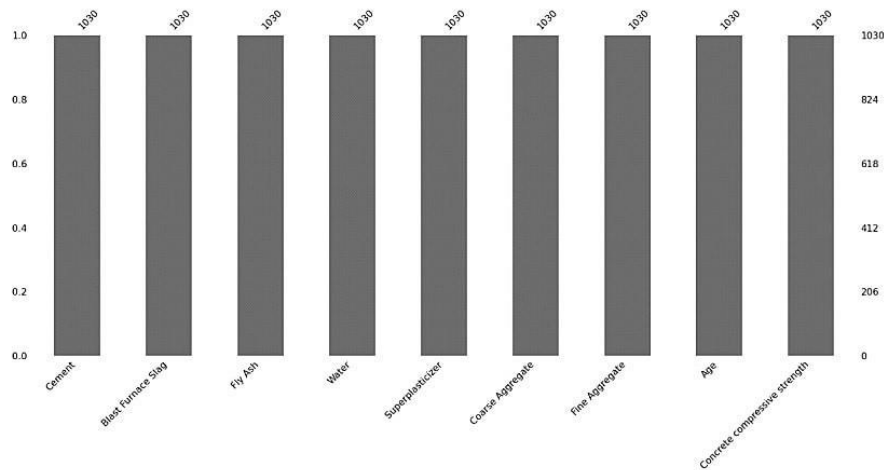
```
Out[16]: Cement                False
Blast Furnace Slag            False
Fly Ash                      False
Water                        False
Superplasticizer             False
Coarse Aggregate             False
Fine Aggregate               False
Age                          False
Concrete compressive strength False
dtype: bool
```

```
In [17]: data.isnull().sum() #used for finding the null values
```

```
Out[17]: Cement                0
Blast Furnace Slag            0
Fly Ash                      0
Water                        0
Superplasticizer             0
Coarse Aggregate             0
Fine Aggregate               0
Age                          0
Concrete compressive strength 0
dtype: int64
```

```
In [19]: sns.heatmap(data.isnull(),cbar=False)
```

```
In [14]: import missingno as msno #finding missing values
msno.bar(data)
plt.show()
```

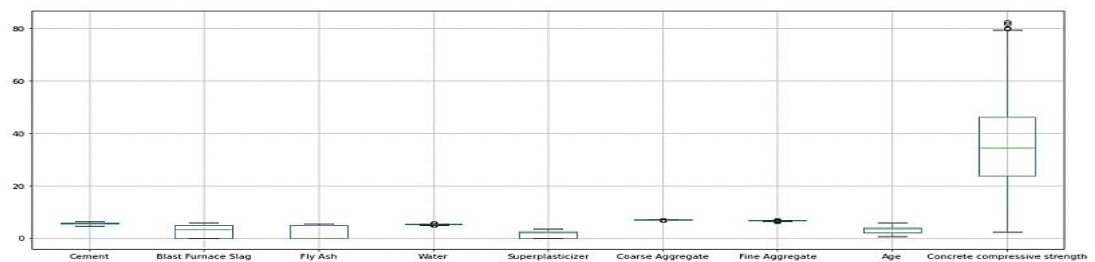


DATA VISUALISATION

Checking with Outliers

```
In [16]: data.boxplot(figsize=(18,7))
```

```
Out[16]: <AxesSubplot: >
```



```
In [17]: def outliers():
    for i in data.columns:
        print("\033[1m" + i + "\033[0m")
        Q1 = data[i].quantile(q=0.25) # 25% quartile value
        Q3 = data[i].quantile(q=0.75) # 75% quartile value
        print('1st Quartile (Q1) is: ', Q1)
        print('3st Quartile (Q3) is: ', Q3)
        IQR = Q3 - Q1 #finding Inter Quartile range
        print(IQR)
        UL = Q3 + (1.5 * IQR) #finding the upper level point

        print("upper limit : ",UL)
        LL = Q1 - (1.5 * IQR) #finding the lower level point
        print("lower limit : ",LL)
        print("Outlier present above UL",data[i][data[i]>UL].count())
        print("Outlier present above LL",data[i][data[i]<LL].count())
        print('---'*40)
    outliers() # calling the function
```

Cement

```
1st Quartile (Q1) is:  5.26462568244329
3st Quartile (Q3) is:  5.860786223465865
0.5961605410225754
upper limit :  6.755027034999729
lower limit :  4.370384870909427
Outlier present above UL 0
Outlier present above LL 0
```

Blast Furnace Slag

```
1st Quartile (Q1) is:  0.0
3st Quartile (Q3) is:  4.969465836003737
4.969465836003737
upper limit :  12.423664590009341
lower limit :  -7.454198754005605
Outlier present above UL 0
Outlier present above LL 0
```

Fly Ash

```
1st Quartile (Q1) is:  0.0
3st Quartile (Q3) is:  4.78164132910387
4.78164132910387
upper limit :  11.954103322759675
lower limit :  -7.172461993655805
Outlier present above UL 0
Outlier present above LL 0
```


Water

1st Quartile (Q1) is: 5.111385197196399
3rd Quartile (Q3) is: 5.262690188904886
0.15130499170848655
upper limit : 5.489647676467616
lower limit : 4.88442770963367
Outlier present above UL 2
Outlier present above LL 12

Superplasticizer

1st Quartile (Q1) is: 0.0
3rd Quartile (Q3) is: 2.4159137783010487
2.4159137783010487
upper limit : 6.0397844457526215
lower limit : -3.6238706674515733
Outlier present above UL 0
Outlier present above LL 0

Coarse Aggregate

1st Quartile (Q1) is: 6.838405200847344
3rd Quartile (Q3) is: 6.93770235535009
0.09929715450274568
upper limit : 7.086648087104209
lower limit : 6.689459469093226
Outlier present above UL 0
Outlier present above LL 6

Fine Aggregate

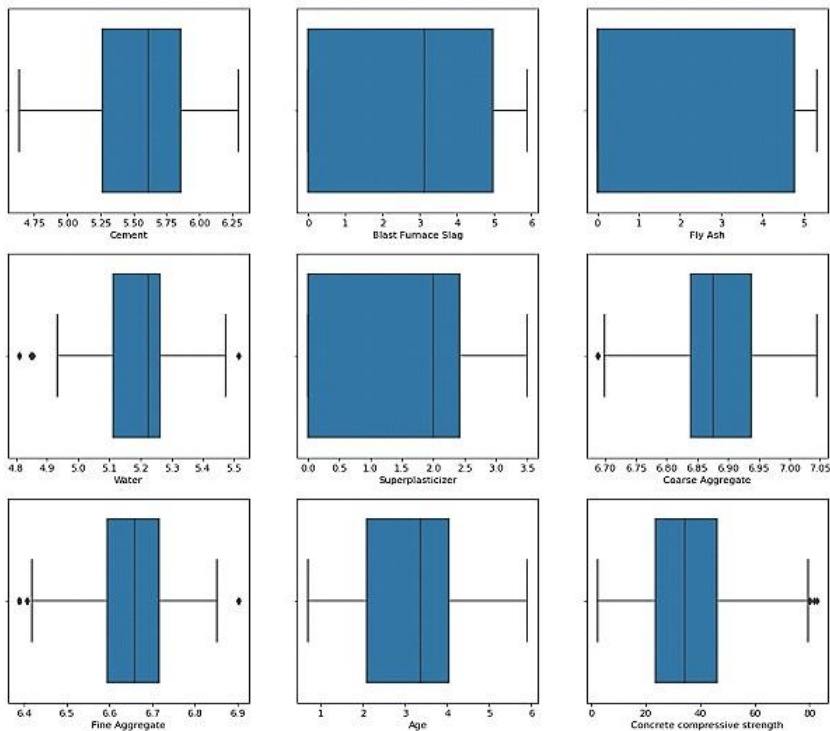
1st Quartile (Q1) is: 6.595711359522024
3rd Quartile (Q3) is: 6.715383386334681
0.11967202681265654
upper limit : 6.894891426553666
lower limit : 6.416203319303039
Outlier present above UL 5
Outlier present above LL 35

Age

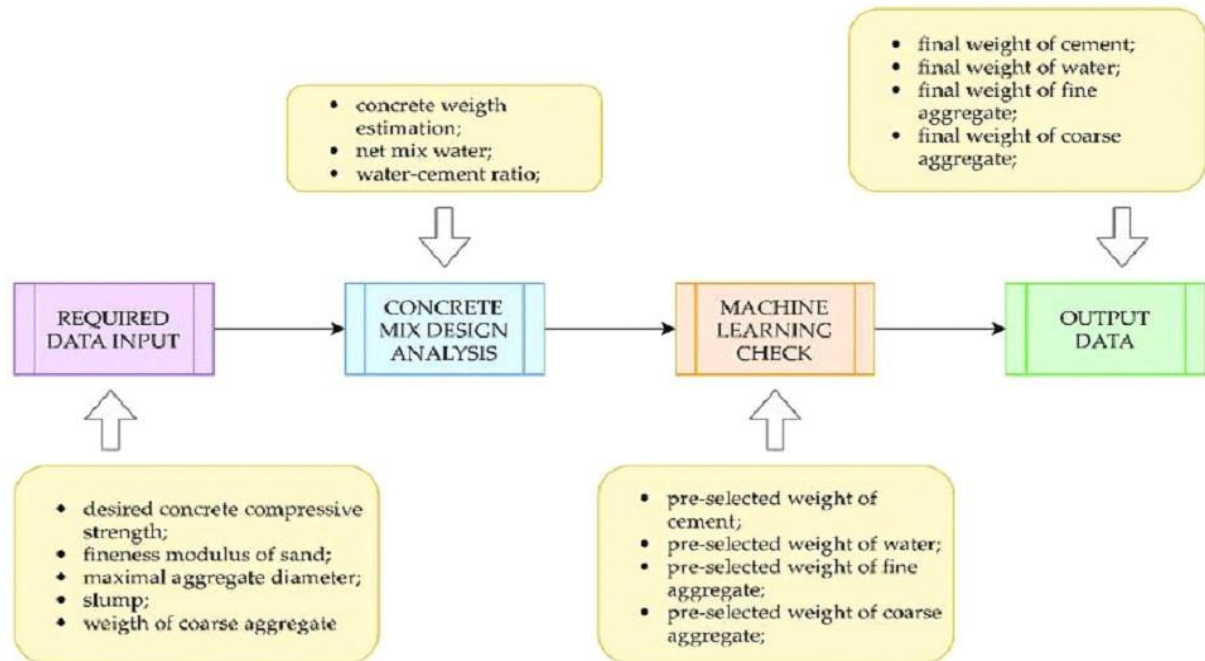
1st Quartile (Q1) is: 2.0794415416798357
3rd Quartile (Q3) is: 4.04305126783455
1.9636097261547145
upper limit : 6.988465857066622
lower limit : -0.8659730475522358
Outlier present above UL 0
Outlier present above LL 0

Concrete compressive strength

1st Quartile (Q1) is: 23.709999999999997



4.FLOWCHART



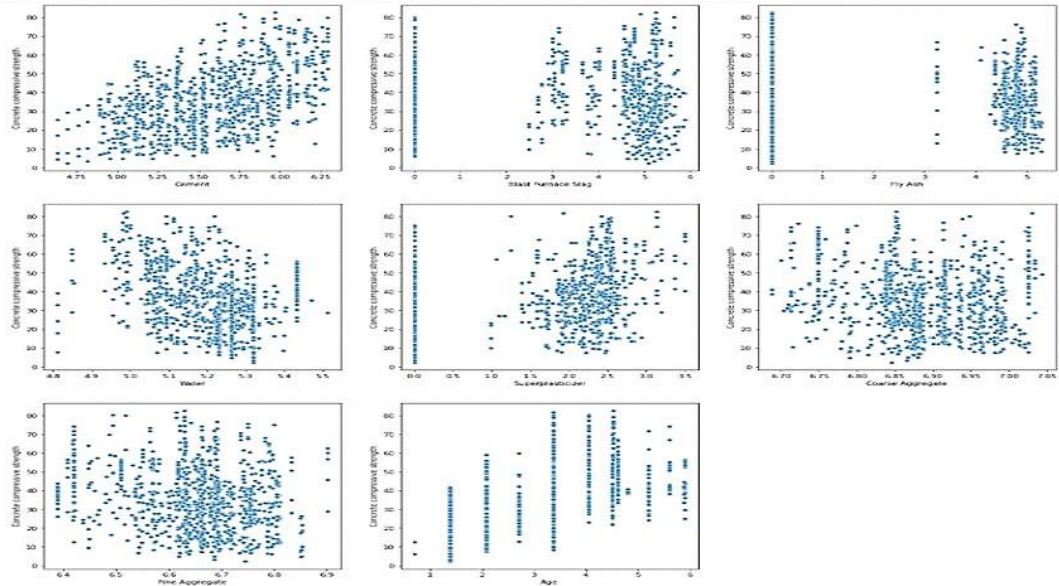
6.RESULT

ANACONDA (JUPYTER)

Scatterplot with the Target Column

```
In [30]: # y=data["Concrete compressive strength"] #target column
plt.figure(figsize=(20,20),facecolor='white') #setting the figure size
plotnumber = 1

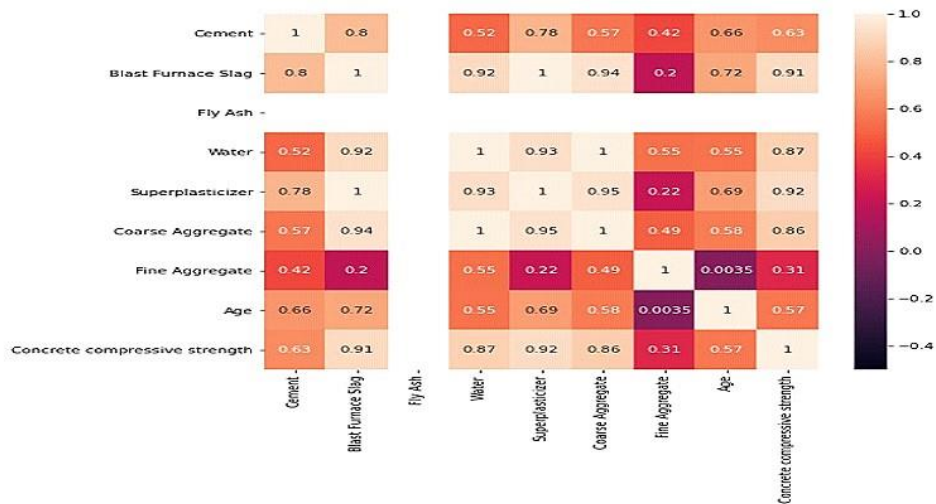
for column in data.columns[:8]: #iterating through each columns
    ax = plt.subplot(3,3,plotnumber) #as we have 9 columns that why we are 3X3
    sns.scatterplot(data[column],y) #plotting scatterplot
    plt.xlabel(column,fontsize=10) #labeling the column name at x-axis
    plotnumber+=1
plt.show()
```



Correlation between the independent columns

```
In [31]: # plt.figure(figsize=(8,8)) #figure size of width-8 and height-8
sns.heatmap(data[:8].corr().abs(), vmin = -0.5,vmax = 1,annot=True)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x261bf172220>



Creating the Independent and Dependent Columns

```
In [21]: # x=pd.DataFrame(data,columns=data.columns[:8]) #independent columns
y=pd.DataFrame(data,columns=data.columns[8:]) #dependent column
```

Splitting the data into train and split

```
In [22]: # x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
print(x_train.shape)
print(y_train.shape)

(824, 8)
(824, 1)
```

Training and testing the model

```
In [22]: from sklearn.ensemble import GradientBoostingRegressor
mr=GradientBoostingRegressor() #object of GradientBoostingRegressor
mr.fit(x_train,y_train)

C:\Users\DELL\Anaconda3\lib\site-packages\sklearn\ensemble\gradient_boosting.py:1450: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

Out[22]: GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
learning_rate=0.1, loss='ls', max_depth=3,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
```

```
In [23]: y_pred=mr.predict(x_test)
print("prediction made by Gradient Boosting model:",y_pred)

prediction made by Gradient Boosting model: [21.95234148 13.08162009 24.70391082 11.32025646 33.85349972 64.77465613
12.45041903 19.73265122 40.42710467 30.13460115 20.64180637 14.72181082
37.33435548 31.47690119 8.39548073 28.19448467 38.81451032 60.80235308
55.41349105 36.17514766 22.09653922 50.34586419 17.54229031 59.22520866
69.79200088 30.18414407 23.47870871 29.62185842 35.82572885 14.74206084
42.2242261 54.78728341 27.06673021 24.87332946 34.92824549 39.36193638
16.50942869 26.70660935 25.92994908 14.78114178 39.53782087 13.46243396
47.04232237 38.01291844 35.78935294 27.67350801 33.28586165 27.08849056
34.08139794 51.48267971 25.15450245 35.37386967 61.10296174 53.1944813
56.51847802 8.85042266 38.02781274 22.95231442 32.99320901 44.64324648
49.32434581 41.57929522 52.63226021 49.50361013 22.93940688 34.73179197
48.47116198 35.05219693 40.66955402 27.65021804 29.83790459 27.00297825
39.36193638 12.80090167 67.55248442 58.04726327 53.09303053 14.88848036
60.3385503 42.22527746 37.64200033 44.22126966 49.57778314 33.26437738
45.31391031 41.93153053 33.58740541 60.52581393 49.57778314 49.71024993
33.9313478 36.22609241 28.20107518 40.04633608 31.31587817 68.47187226
38.50036265 12.5517226 31.74158529 28.59089539 35.97574895 61.51911155
36.17473014 41.02750570 46.5706706 36.20474061 26.76246120 37.07600354
16.49113295 23.60374375 23.2620048 35.29585805 9.41309786 27.67350801
32.39097473 16.33648516 36.37686244 40.46295946 46.66434763 56.19940098
76.01180104 57.01508663 38.50036265 19.37608445 13.40745862 28.41321305
54.95947943 14.30626622 69.92003204 21.70176448 36.19854895 32.73741861
14.54489222 33.89042188 38.86275688 40.16331248 32.30097473 38.27628855
34.27967922 11.95881816 52.54721058 38.0769414 35.39508201 44.19367588
66.22672154 47.68704002 40.16331248 16.98263838 34.92824549 44.32035003
67.90656588 47.17307745 80.71643822 45.72714702 32.41107987 29.63765677
27.08849056 35.10386618 54.88493496 51.76906215 26.74302967 48.89461558
47.37026737 22.90851667 59.46738753 43.63203337 44.66027029 32.08914728
37.14901501 23.61010371 38.12198157 13.76147006 43.76526343 30.33367865
47.04232237 61.75066787 45.74176051 42.06965495 33.51071773 79.33920221
47.14491171 62.50829954 44.49112444 48.67994380 28.66081302 49.97134441
5.87053159 24.398099138 30.20971064 34.07030098 72.88888085 24.48121141
27.61685041 39.11765922 10.04541508 55.11080044 36.53168157 28.41321395
28.02038331 38.01291844 36.76246037 46.27485185 6.15096776 36.76246037
35.15677734 40.0706007 ]
```

Model Evaluation

```
In [24]: score=mr.score(x_test,y_test)
print("score of Gradient Boosting model:",score)

score of Gradient Boosting model: 0.8856333219046403
```

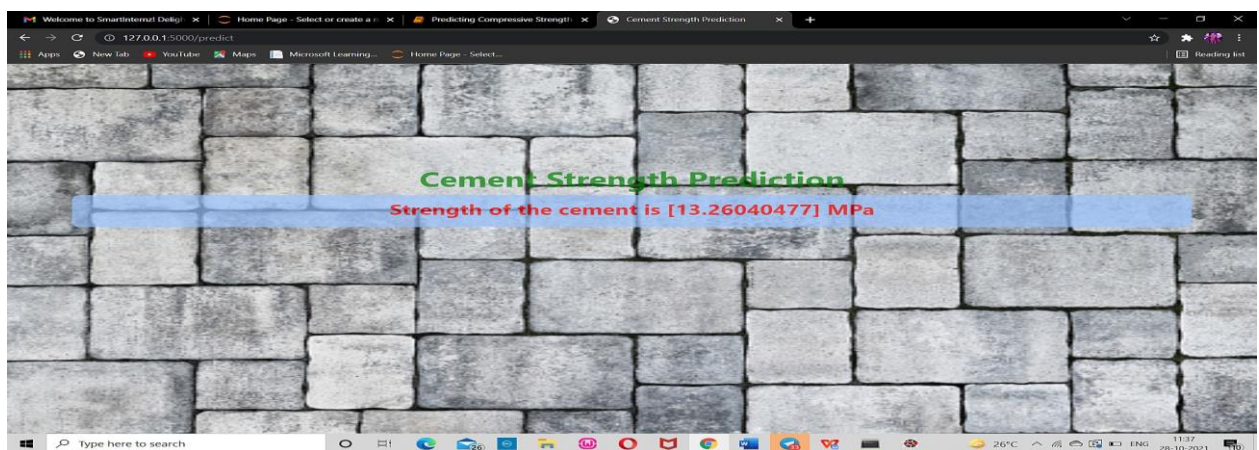
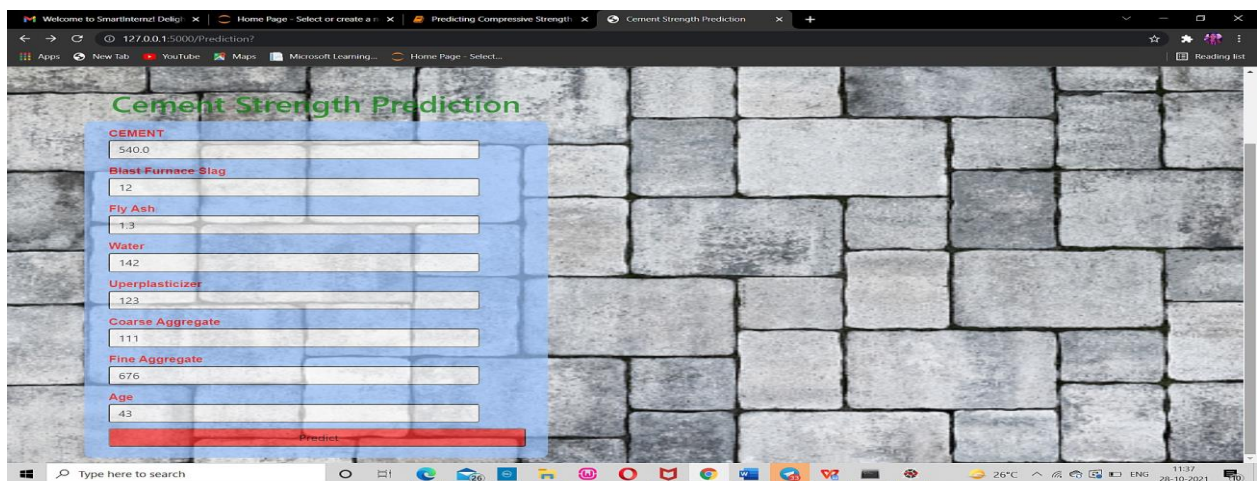
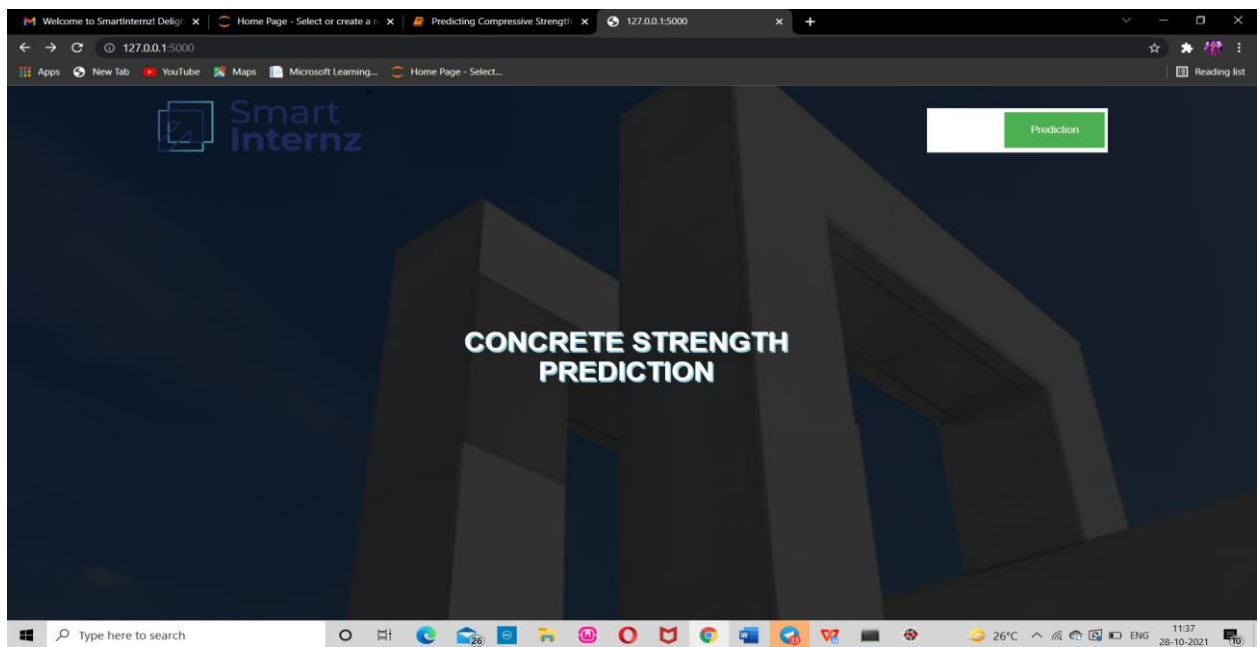
```
In [25]: from sklearn import metrics
print("MAE:",metrics.mean_absolute_error(y_test,y_pred))
print("MSE:",metrics.mean_squared_error(y_test,y_pred))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))

MAE: 3.8428863761561494
MSE: 29.814773086246525
RMSE: 5.460290567931942
```

Dumping our model in pickle format

```
In [26]: pickle.dump(mr, open('cement.pkl','wb'))
```

```
In [ ]:
```

7.ADVANTAGES & DISADVANTAGES

Advantages:

Over the past five years, the implementations of the non-tuned machine learning model have shown a noticeable progress in multidisciplinary of science and engineering fields. This is due to its advantages (particularly over conventional artificial neural network algorithms) such as the randomly initiated hidden neurons without the need for iterative tuning process for free parameters or connections between hidden and output layer.

It is remarkably efficient to reach a global optimum, following universal approximation capability of single layer feed-forward network. Also, this model is featured by the efficiency and generalization performance over traditional learning algorithms (e.g., SVMs or ANNs) as revealed in the estimation problems in many different fields.

With the development of artificial intelligence, it is easier to forecast concrete compressive strength. When compared with other traditional regression methods, machine learning adopts specific algorithms that can learn from the input data and gives highly accurate results.

Disadvantages:

Concrete compressive strength requirements can vary from 2500 psi (17MPa) for residential concrete to 4000 psi (28 MPa) and higher in commercial structures. Higher strengths up to and exceeding 10000 psi (70 MPa) are specified for certain applications.

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

Any disadvantage of using a multiple regression model usually comes down to the data being used. Two examples of this are using incomplete data and falsely concluding that a correlation is a causation.

8.APPLICATIONS

Concrete is considered by many to be a strong and durable material, and rightfully so. But there are different ways to assess concrete strength.

Water/cementitious ratio (w/cm)

This refers to the ratio of water to cement in the concrete mixture. A lower water to cement ratio makes for a stronger concrete, but it also makes the concrete more difficult to work with. The right balance must be struck to achieve the desired strength while maintaining workability.

Proportioning

Traditional concrete is made of water, cement, air, and an aggregate mixture of sand, gravel, and stone. The right proportion of these ingredients is key for achieving a higher concrete strength.

A concrete mixture with too much cement paste may be easy to pour—but it will crack easily and not withstand the test of time. Conversely, too little cement paste will yield a concrete that is rough and porous.

Mixing

Optimal mixing time is important for strength. While strength does tend to increase with mixing time to a certain point, mixing for too long can actually cause excess water evaporation and the formation of fine particles within the mix. This ends up making the concrete harder to work with and less strong.

There is no golden rule for optimal mixing time, as it depends on many factors, such as: the type of mixer being used, the speed of the mixer rotation, and the specific components and materials within a given batch of concrete.

Curing methods

The longer the concrete is kept moist, the stronger it will become. To protect the concrete, precautions must be taken when curing concrete in extremely cold or hot temperatures.

9.CONCLUSION

Earlier and accurate estimation of concrete strength are valuable to the construction industry. The presence of such model would possibly obtain the hard balance and equality between controlling the quality (quality control process) and economics (saving time and expense, i.e., this model could be used in construction to make the necessary adjustments on mix proportion used, to avoid situations where, concrete does not reach the required design strength or by avoiding concrete that is unnecessarily strong.

This methodology allows a fast and accurate prediction of values for compressive strength on site. Common methods for estimation of in place strength requires extensive use of curing of mortar cubes at constant temperatures or the use of databases containing a large number of compressive strength values made at many ages and cured at different temperatures. These databases have to be fed with a statistical relevant amount of data before a reliable estimation of the strength can be made. Furthermore, all of these methods requires many hours of lab and field time for testing, collecting and analysing data.

Furthermore, the existing variables in the model yielded good reasonable results. Also, it is not preferred to load the prediction model with large number of variables, because it is preferred to use a model with lesser number of variables with higher possible accuracy to assure the rapid and easy use of the model.

10. FUTURE SCOPE

Compressive strength is one of the most important properties of concrete and mortar. The strength of the binder (cement) therefore has a significant effect on the performance characteristics of the mixture and ensures the overall quality of the finished product. The test for compressive strength is generally carried out by crushing cubes of hardened cement-sand mortar in a compression machine.

For now, we have created the web application of the process, which can be further converted to an android app which will be more useful and convenient for the user to use and will help in saving time.

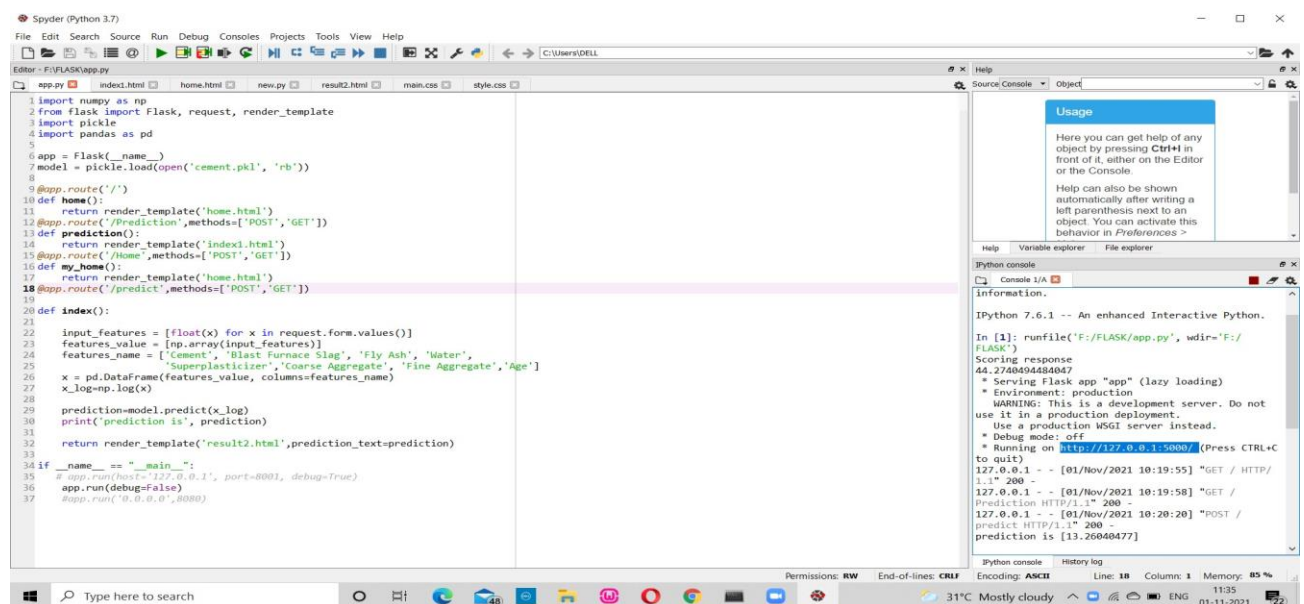
App development will help in creating the project process oriented.

11. BIBLIOGRAPHY

1. Comprehensive Machine Learning-Based Model for Predicting Compressive Strength of Ready-Mix Concrete
 2. PREDICTING COMPRESSIVE STRENGTH OF CONCRETE FOR VARYING WORKABILITY USING REGRESSION MODELS
- Article in International Journal Of Engineering & Applied Sciences · December 2014

APPENDIX

a. Source Code



```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4 import pandas as pd
5
6 app = Flask(__name__)
7 model = pickle.load(open('cement.pkl', 'rb'))
8
9 @app.route('/')
10 def home():
11     return render_template('home.html')
12 @app.route('/prediction', methods=['POST', 'GET'])
13 def prediction():
14     return render_template('index1.html')
15 @app.route('/home', methods=['POST', 'GET'])
16 def my_home():
17     return render_template('home.html')
18 @app.route('/predict', methods=['POST', 'GET'])
19
20 def index():
21     input_features = [float(x) for x in request.form.values()]
22     features_value = [np.array(input_features)]
23     features_name = ['Cement', 'Blast Furnace Slag', 'Fly Ash', 'Water',
24                     'Superplasticizer', 'Coarse Aggregate', 'Fine Aggregate', 'Age']
25     x = pd.DataFrame(features_value, columns=features_name)
26     x_log=np.log(x)
27
28     prediction=model.predict(x_log)
29     print('prediction is', prediction)
30
31     return render_template('result2.html', prediction_text=prediction)
32
33 if __name__ == '__main__':
34     # app.run(host='127.0.0.1', port=8080, debug=True)
35     app.run(debug=False)
36
37 app.run(host='0.0.0.0', port=8080)
```


b.UI output Screenshots

