PREDICTING COMPRESSIVE STRENGTH OF CONCRETE USING IBM WATSON MACHINE LEARNING

A PROJECT REPORT

Submitted in partial fulfilment for the data science externship

with

SmartInternz

by

NAME	REGITRATION NO.
Rishika	18BEI0119
Simran Rajpal	18BEI0123
Kshitij Dubey	18BEI0128
Pragya Jain	18BEE0253

Under the guidance of

SmartInternz Team

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

INTRODUCTION

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.

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).

LITERATURE SURVEY

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.

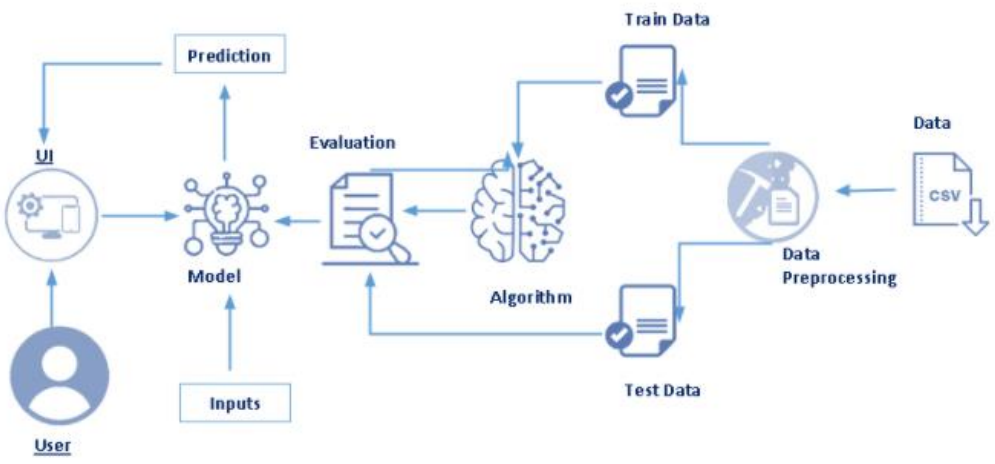
For the experimental data obtained, the data sets on the dependent variable were normally distributed for each of the possible combinations of the level of the X variables, it was deemed suitable to use regression models for prediction of concrete strengths. Bayrak and Akgül predicted the lifetime performance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

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.

THEORITICAL ANALYSIS

Block Diagram



Hardware/ Software designing

https://github.com/rishika28/Applied-Data-Science/blob/main/Predicting%20Compressive%20Strength%20of%20Concrete%20(1).ipynb

EXPERIMENTAL INVESTIGATIONS

DATA PREPROCESSING

Importing Libraries

In [1]:
```
import pandas as pd #used for data manipulation
import numpy as np #used for numerical analysis
from collections import Counter as c # return counts
import seaborn as sns #used for data Visualization
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split #splits data in random train and test array
from sklearn.metrics import accuracy_score,mean_squared_error,mean_absolute_error#model performance
import pickle #Python object hierarchy is converted into a byte stream,
from sklearn.linear_model import LinearRegression #Regresssion ML algorithm
```

Loading Dataset

In [2]:
```
data=pd.read_csv(r"D:\concrete.csv") #loading the csv dataset
```

In [4]:
```
data.head() #returns first 5 rows of data
```

Out[4]:

	Cement (component 1) (kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3) (kg in a m^3 mixture)	Water (component 4) (kg in a m^3 mixture)	Superplasticizer (component 5)(kg in a m^3 mixture)	Coarse Aggregate (component 6) (kg in a m^3 mixture)	Fine Aggregate (component 7) (kg in a m^3 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

Seeing Target, Nominal and Numerical Columns Count

In [14]:
```
np.unique(data.dtypes,return_counts=True)
```

Out[14]:
```
(array([dtype('int64'), dtype('float64')], dtype=object),
 array([1, 8], dtype=int64))
```

In [15]:
```
data.dtypes[data.dtypes!='O'].index.values
```

Out[15]:
```
array(['Cement', 'Blast Furnace Slag', 'Fly Ash', 'Water',
       'Superplasticizer', 'Coarse Aggregate', 'Fine Aggregate', 'Age',
       'Concrete compressive strength'], dtype=object)
```

Null Values

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

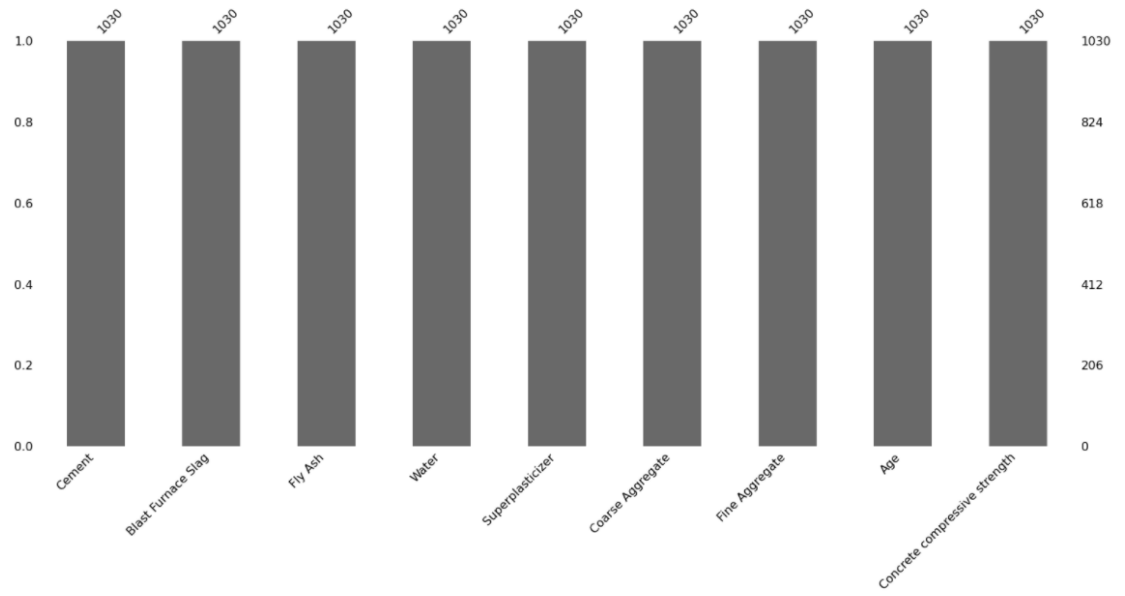
Out[16]:
```
Cement                          False
Blast Furnace Slag              False
Fly Ash                         False
Water                           False
Superplasticizer                False
Coarse Aggregate                False
Fine Aggregate                  False
Age                             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()
```

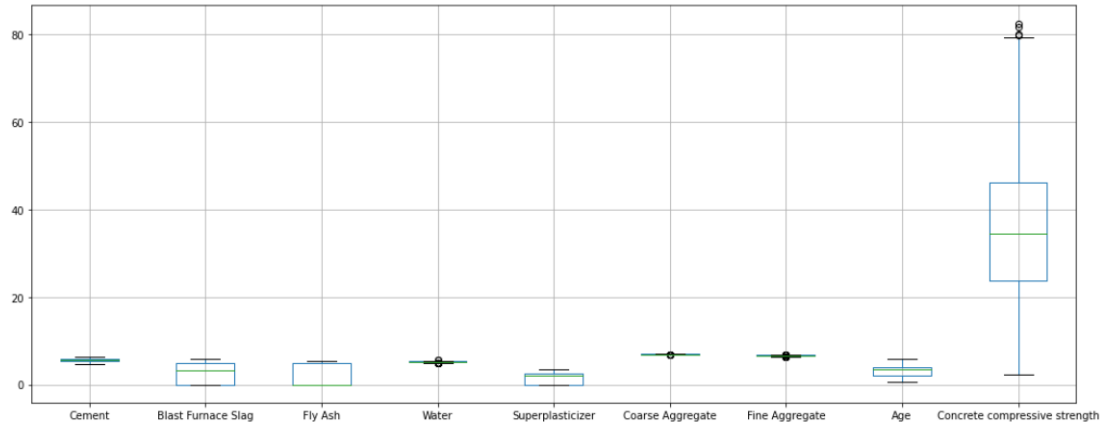


DATA VISUALISATION

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
```

Water
1st Quartile (Q1) is: 5.111385197196399
3st 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

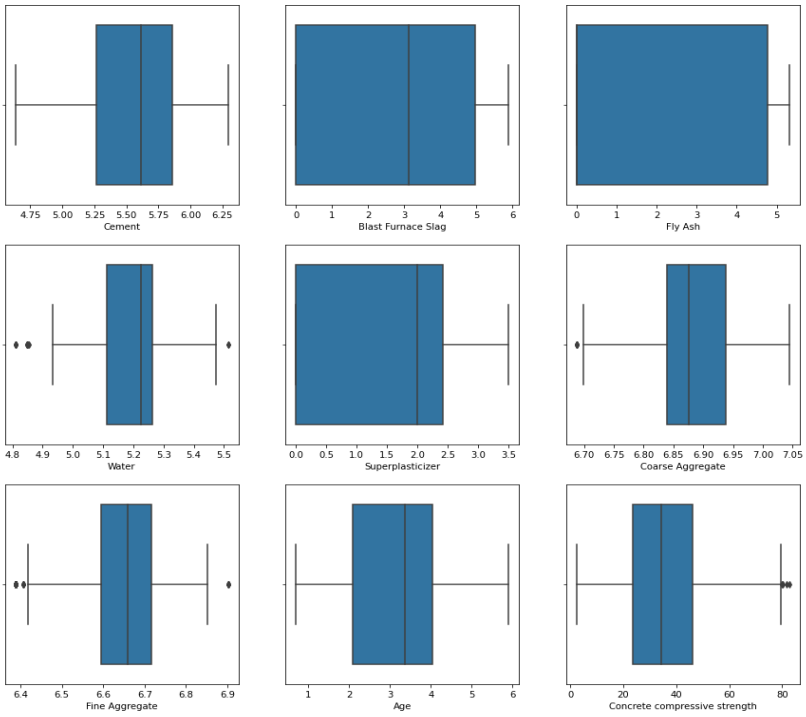
Superplasticizer
1st Quartile (Q1) is: 0.0
3st Quartile (Q3) is: 2.4159137783010487
2.4159137783010487
upper limit : 6.0397844457526215
lower limit : -3.6238706674515733
Outlier present above UL 0
Outlier present above LL 0

Coarse Aggregate
1st Quartile (Q1) is: 6.838405200847344
3st 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

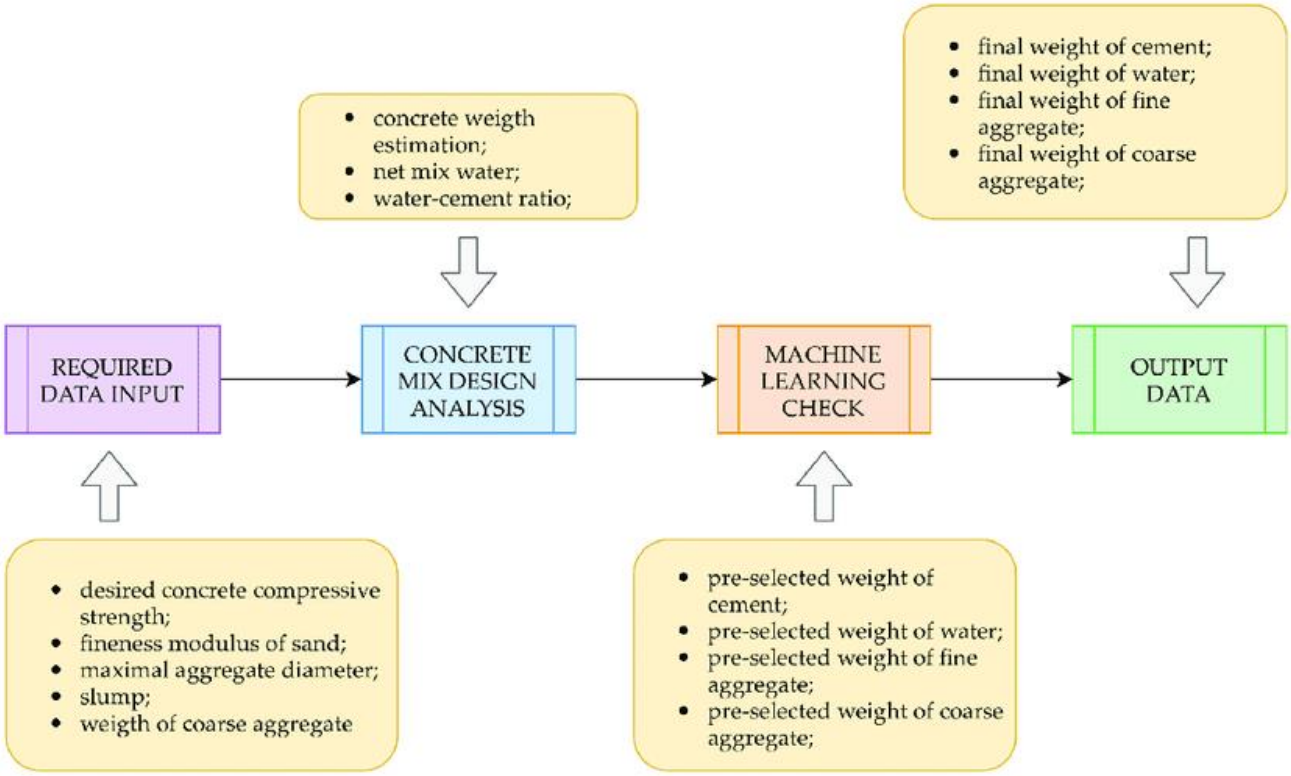
Fine Aggregate
1st Quartile (Q1) is: 6.595711359522024
3st Quartile (Q3) is: 6.715383386334681
0.11967202681265654
upper limit : 6.894891426553666
lower limit : 6.416203319303039
Outlier present above UL 5
Outlier present above LL 35

Age
1st Quartile (Q1) is: 2.0794415416798357
3st 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



FLOWCHART


final weight of cement;
final weight of water;
final weight of fine aggregate;
final weight of coarse aggregate;
concrete weigth estimation;
net mix water;
water-cement ratio;
REQUIRED DATA INPUT
CONCRETE MIX DESIGN ANALYSIS
MACHINE LEARNING CHECK
OUTPUT DATA
desired concrete compressive strength;
fineness modulus of sand;
maximal aggregate diameter;
slump;
weigth of coarse aggregate
pre-selected weight of cement;
pre-selected weight of water;
pre-selected weight of fine aggregate;
pre-selected weight of coarse aggregate;

RESULT

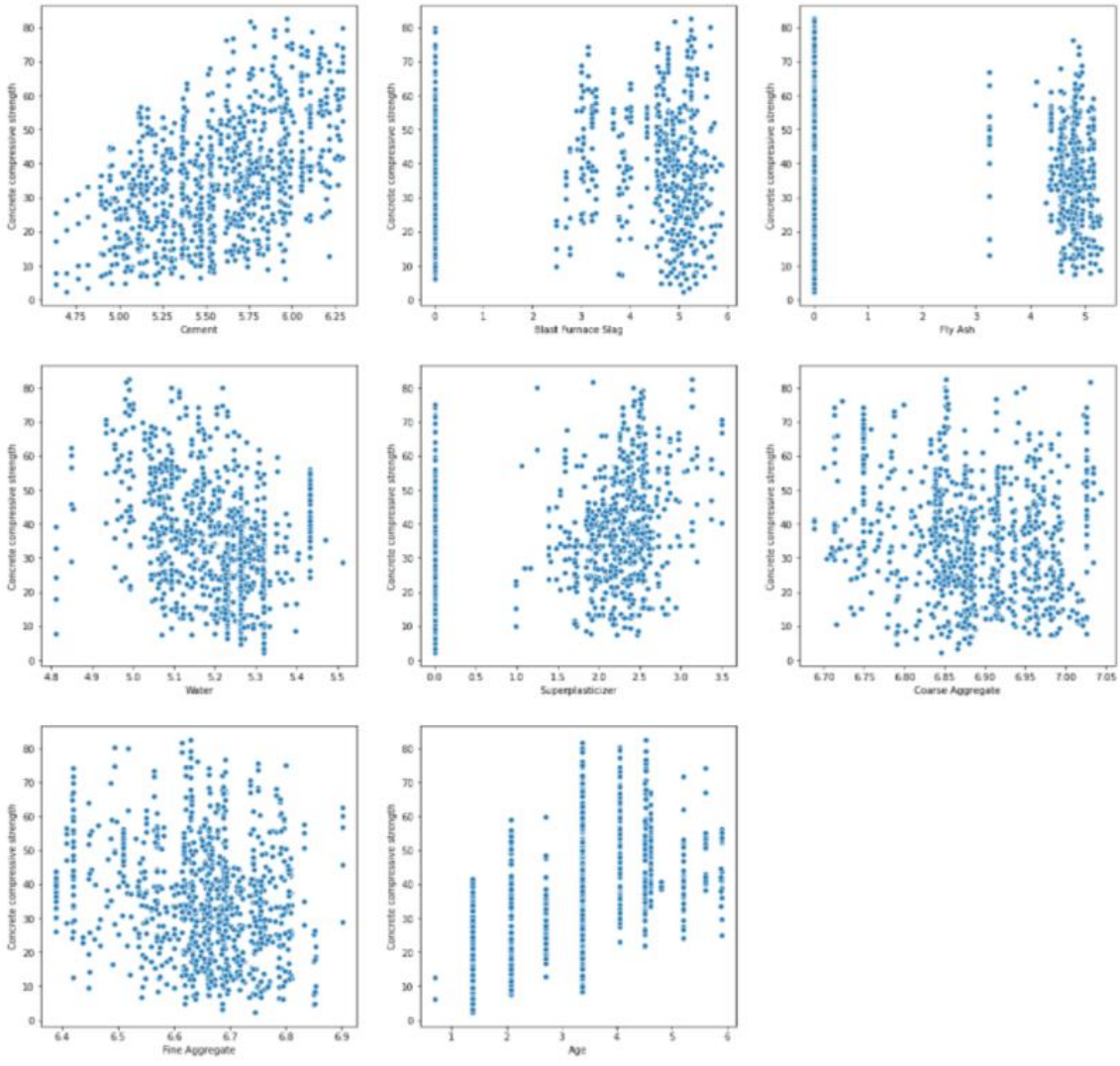
ANACONDA (JUPYTER)

Scatterplot with the Target Column

In [30]:

```
y1=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],y1) #plottng 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]) #independet 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)
```

Gradient Boosting Regressor

In [23]:
```
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor() #object of GradientBoostingRegressor
gbr.fit(x_train,y_train) # training our data using fit method
```

```
C:\anaconda\lib\site-packages\sklearn\utils\validation.py:72: 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().
  return f(**kwargs)
```

Out[23]:
```
GradientBoostingRegressor()
```

Checking the score with both the model

In [24]:
```
score=gbr.score(x_test,y_test)

print("Score of Gradient Boosting model:",score)
```

```
Score of Gradient Boosting model: 0.8856680197926448
```

Predicting the output with our build models

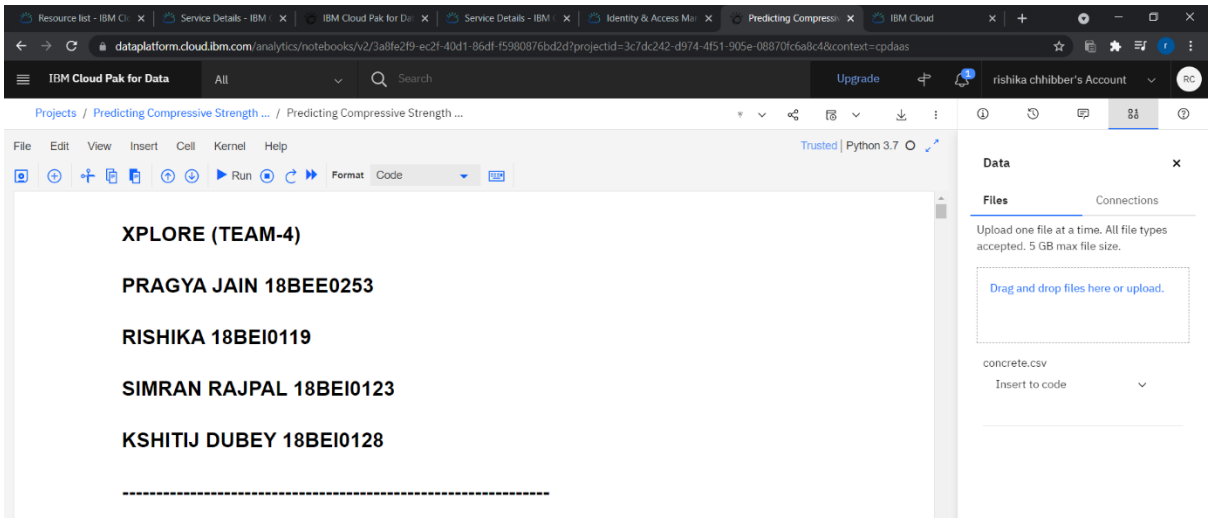
In [25]:
```
y_pred =gbr.predict(x_test) #predicting using predict method
print("Prediction made by Gradient Boosting model:",y_pred)
```

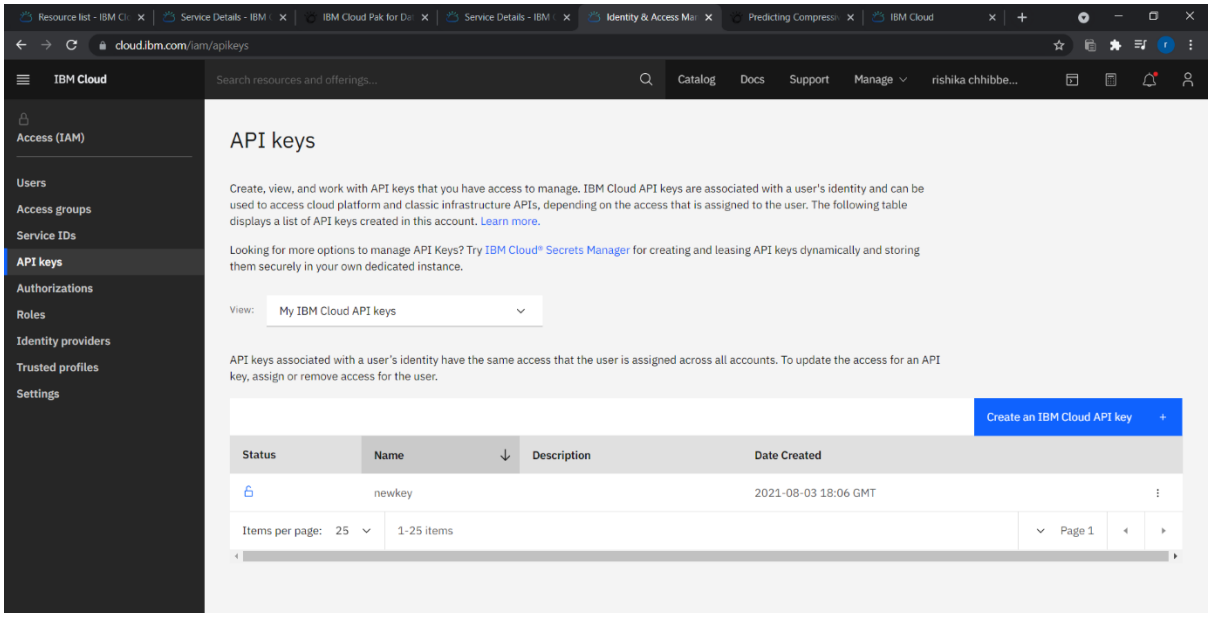
```
Prediction made by Gradient Boosting model: [21.95234148 13.08162009 24.70391082 11.32025646 33.85349972 64.77465613
 18.45044902 48.72305133 40.43710467 20.12460115 38.64180627 14.73181983
 37.33435548 31.47690119  8.39548073 28.19448467 38.81451032 60.80235308
 55.41349105 36.17514766 22.09653922 50.34586419 17.54229031 59.22520866
 60.28299988 39.18419497 33.47870521 29.62105842 35.97574895 13.79206845
 42.2242261  54.78728341 27.06673021 24.87332946 34.92824549 39.36193638
 16.50942869 26.70669935 25.92994908 14.78114178 39.53782087 13.46243396
 47.04232237 38.96167764 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 22.28661078 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  43.22352746 37.64200033 44.22126966 49.57778314 33.2643738
 45.31391031 41.93153053 33.58746541 60.52581393 49.57778314 49.71624993
 33.9313478  36.22609241 28.20107518 40.04633608 31.31587817 68.47187226
 38.50036265 12.5517226  31.74158529 28.59089539 35.97574895 61.51911155
 36.17473814 41.82750579 46.5796786  36.20474961 26.76246129 37.07609354
 16.49113295 23.60374375 23.2620048  35.29585805  9.41309786 27.67350801
 32.39097473 16.33648516 36.37686244 40.46295946 46.66434763 56.19940098
 76.91180194 57.01508663 38.50036265 19.2760845  13.40745862 28.41321395
 54.95947943 14.10626622 69.92003204 21.70176448 16.19854885 32.73741861
 14.54489222 33.89042188 38.86275688 40.16331248 32.39097473 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 26.74362967 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 79.33930221
 27.33493371 62.58029953 34.49132432 38.67994386 28.64681362 39.97154441
  5.87053159 24.39809138 30.20971064 34.07030098 72.88888085 24.48121141
 27.61685941 39.11765922 10.04541508 55.11080044 36.53168157 28.41321395
 28.02038331 38.96167764 36.76346037 46.27485185  6.15696776 36.76346037
 35.15677734 40.0706007 ]
```

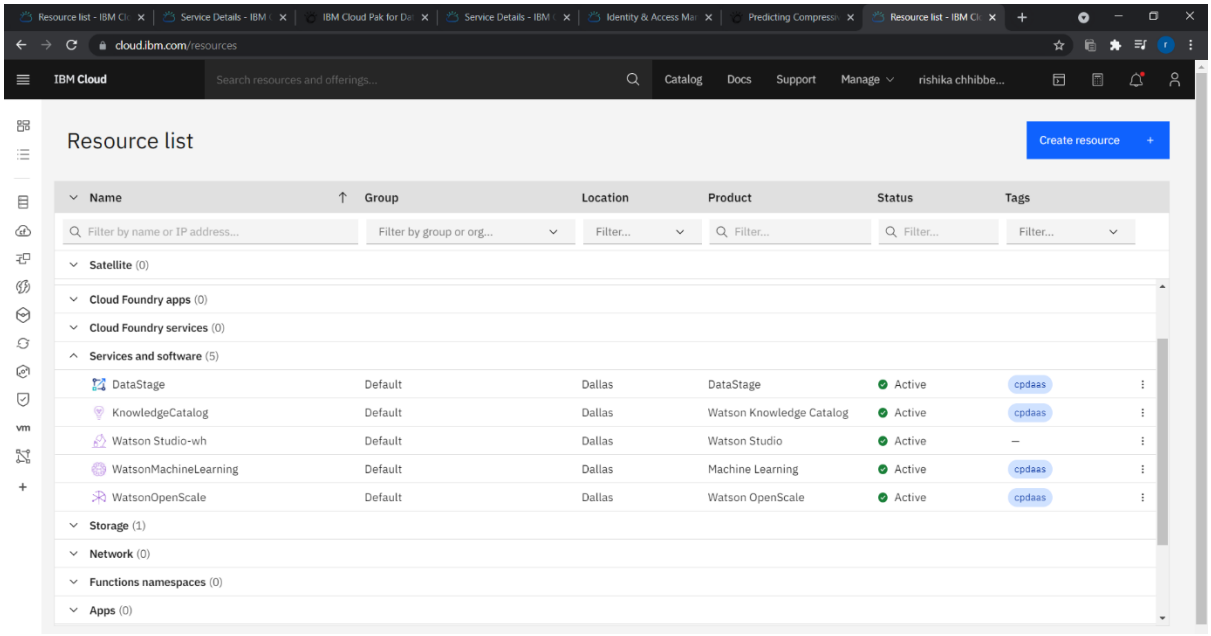
In [26]:
```
from sklearn import metrics #importing the metrics library
print("MAE:",metrics.mean_absolute_error(y_test,y_pred)) #Mean Absolute Error
print("MSE:",metrics.mean_squared_error(y_test,y_pred)) # Mean Square Error
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test,y_pred))) # Root Mean Square Error
```

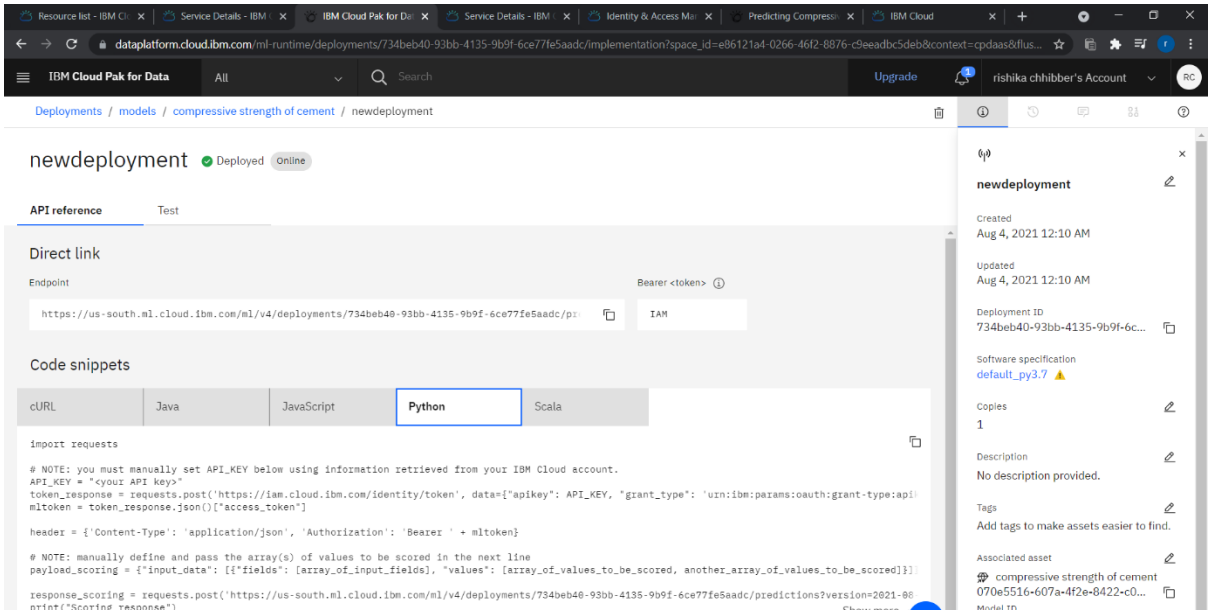
```
MAE: 3.8400642061622556
MSE: 29.805727534914134
RMSE: 5.459462201986028
```

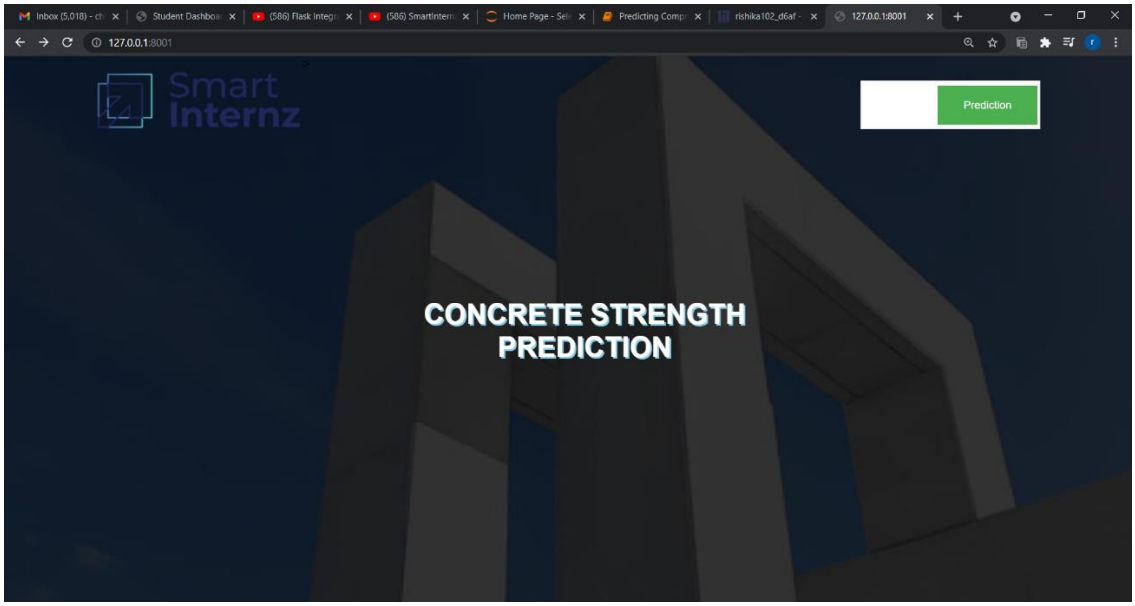
IBM WATSON STUDIO



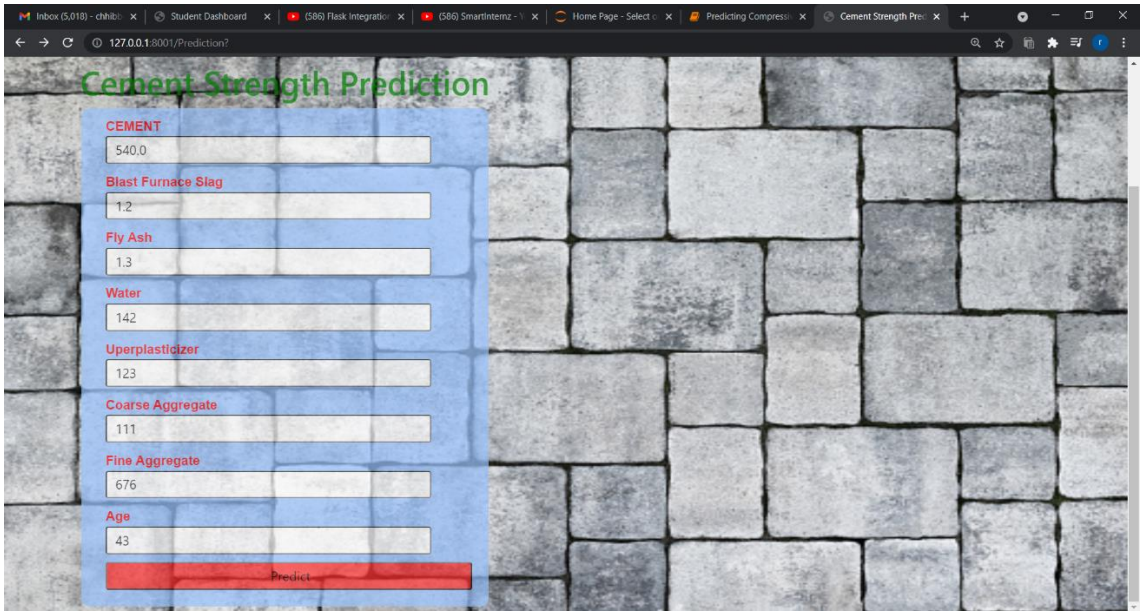



cloud.ibm.com/resources
IBM Cloud
Search resources and offerings...
Catalog
Docs
Support
Manage
rishika chhibbe...
Resource list
Create resource
Name
Group
Location
Product
Status
Tags
Filter by name or IP address...
Filter by group or org...
Satellite (0)
Cloud Foundry apps (0)
Cloud Foundry services (0)
Services and software (5)
DataStage Default Dallas DataStage Active cpdaas
KnowledgeCatalog Default Dallas Watson Knowledge Catalog Active cpdaas
Watson Studio-wh Default Dallas Watson Studio Active –
WatsonMachineLearning Default Dallas Machine Learning Active cpdaas
WatsonOpenScale Default Dallas Watson OpenScale Active cpdaas
Storage (1)
Network (0)
Functions namespaces (0)
Apps (0)


dataplatform.cloud.ibm.com/ml-runtime/deployments/734beb40-93bb-4135-9b9f-6ce77fe5aadc/implementation?space_id=e86121a4-0266-46f2-8876-c9eeadbc5deb&context=cpdaas&flus...
IBM Cloud Pak for Data
All
Search
Upgrade
rishika chhibber's Account
Deployments / models / compressive strength of cement / newdeployment
newdeployment
Deployed
Online
API reference
Test
Direct link
Endpoint
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/734beb40-93bb-4135-9b9f-6ce77fe5aadc/pr
Bearer <token>
IAM
Code snippets
cURL
Java
JavaScript
Python
Scala
import requests
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "<your API key>"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:api
mltoken = token_response.json()["access_token"]
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"fields": [array_of_input_fields], "values": [array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]
response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/734beb40-93bb-4135-9b9f-6ce77fe5aadc/predictions?version=2021-08
print("Scoring response")
newdeployment
Created
Aug 4, 2021 12:10 AM
Updated
Aug 4, 2021 12:10 AM
Deployment ID
734beb40-93bb-4135-9b9f-6c...
Software specification
default_py3.7
Copies
1
Description
No description provided.
Tags
Add tags to make assets easier to find.
Associated asset
compressive strength of cement
070e5516-607a-4f2e-8422-c0...


127.0.0.1:8001
Smart Internz
Prediction
CONCRETE STRENGTH PREDICTION

127.0.0.1:8001/predict
Cement Strength Prediction
Strength of the cement is [68.96999649] MPa


127.0.0.1:8001/Prediction?
Cement Strength Prediction
CEMENT
540.0
Blast Furnace Slag
1.2
Fly Ash
1.3
Water
142
Uperplasticizer
123
Coarse Aggregate
111
Fine Aggregate
676
Age
43
Predict

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.

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.

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.

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.

BIBLIOGRAPHY

1. Multiple Regression Model for Compressive Strength Prediction of High-Performance Concrete
   Article in Journal of Applied Sciences · January 2009
2. Comprehensive Machine Learning-Based Model for Predicting Compressive Strength of Ready-Mix Concrete
3. PREDICTING COMPRESSIVE STRENGTH OF CONCRETE FOR VARYING WORKABILITY USING REGRESSION MODELS

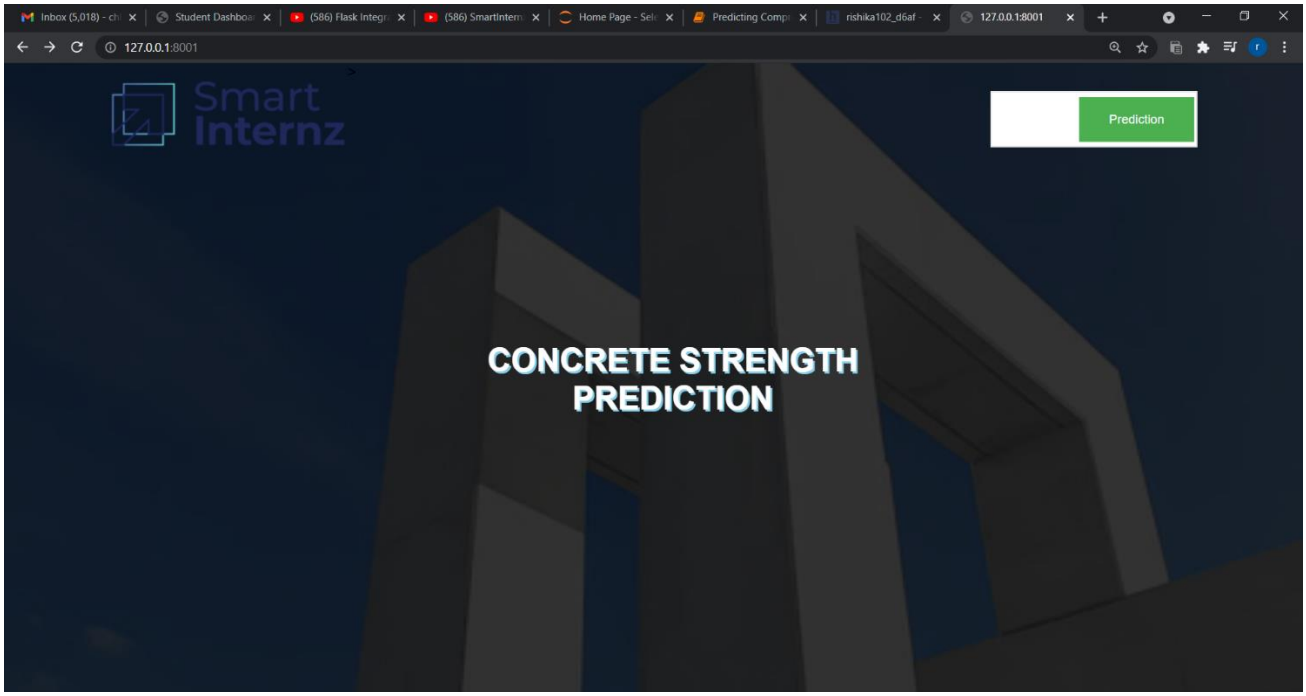
Article in International Journal Of Engineering & Applied Sciences · December 2014
4. https://cor-tuf.com/everything-you-need-to-know-about-concrete-strength/

APPENDIX

a. Source Code

https://github.com/rishika28/Applied-Data-Science/blob/main/Predicting%20Compressive%20Strength%20of%20Concrete%20(1).ipynb

b. UI output Screenshots



127.0.0.1:8001/predict
Cement Strength Prediction
Strength of the cement is [68.96999649] MPa

127.0.0.1:8001/Prediction?
Cement Strength Prediction
CEMENT
540.0
Blast Furnace Slag
1.2
Fly Ash
1.3
Water
142
Uperplasticizer
123
Coarse Aggregate
111
Fine Aggregate
676
Age
43
Predict