

CONCRETE COMPRESSIVE STRENGTH PREDICTION



Project – Machine Learning Technology

Submitted By Paul Praveen



Project Outline

Project Title	Concrete Compressive Strength Prediction
Technologies	Machine Learning Technology
Domain	Infra
Project Difficulties level	Intermediate



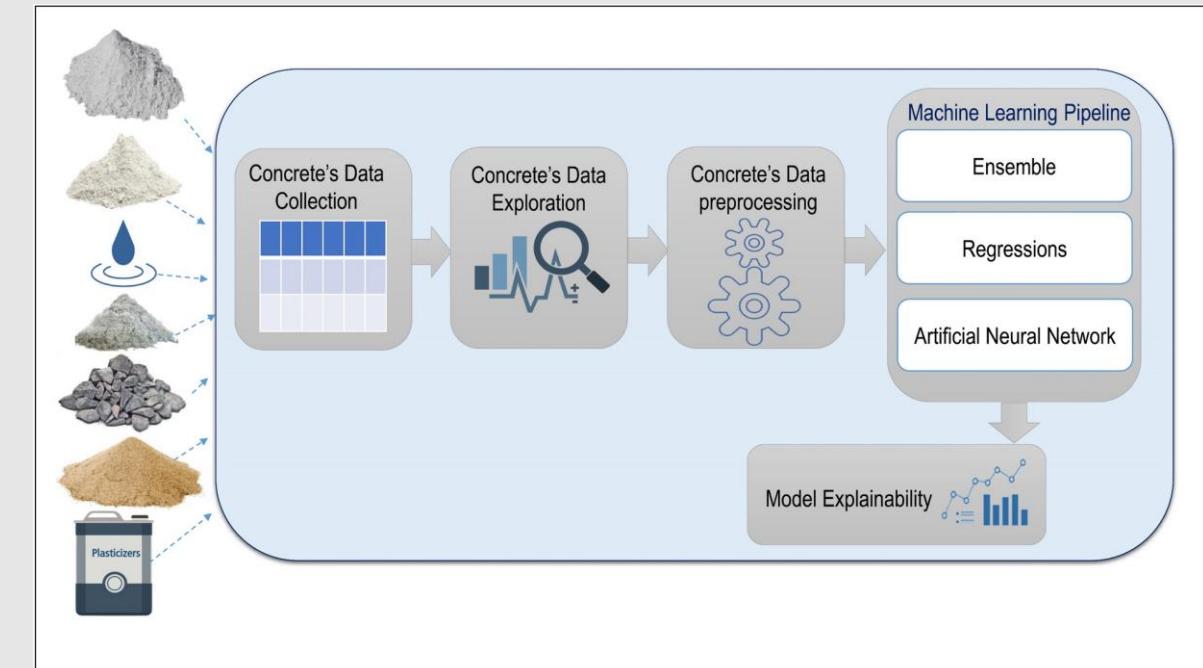
Problem Statement

The quality of concrete is determined by its compressive strength, which is measured using a conventional crushing test on a concrete cylinder.

The strength of the concrete is also a vital aspect in achieving the requisite longevity. It will take 28 days to test strength, which is a long period.

So, what will we do now?

We can save a lot of time and effort by using Data Science to estimate how much quantity of which raw material we need for acceptable compressive strength



Problem Statement

Objective :

To build a solution that should able to predict the compressive strength of the concrete.
(Using Machine Learning)

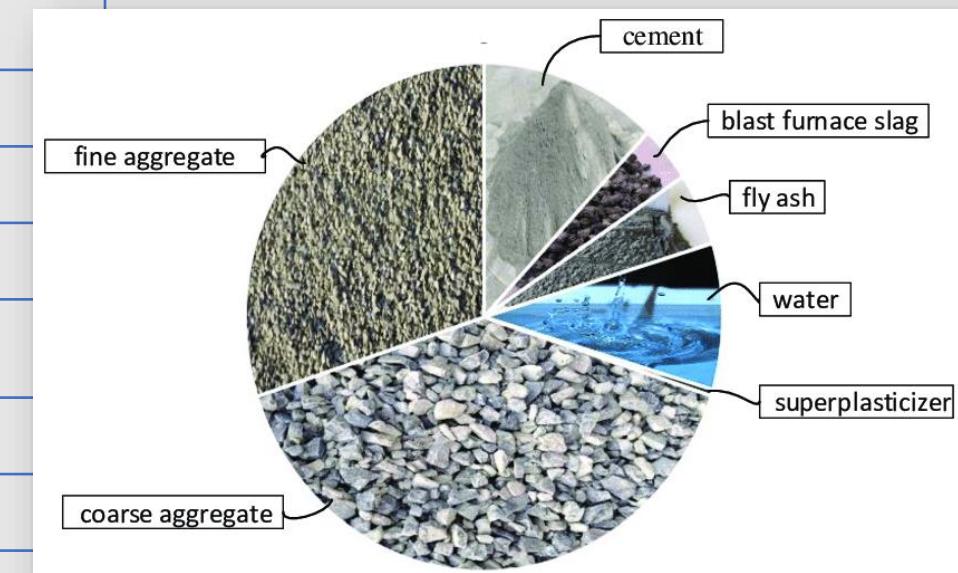
Approach:

The classical machine learning tasks like Data Exploration, Data Cleaning, Feature Engineering, Model Building and Model Testing.

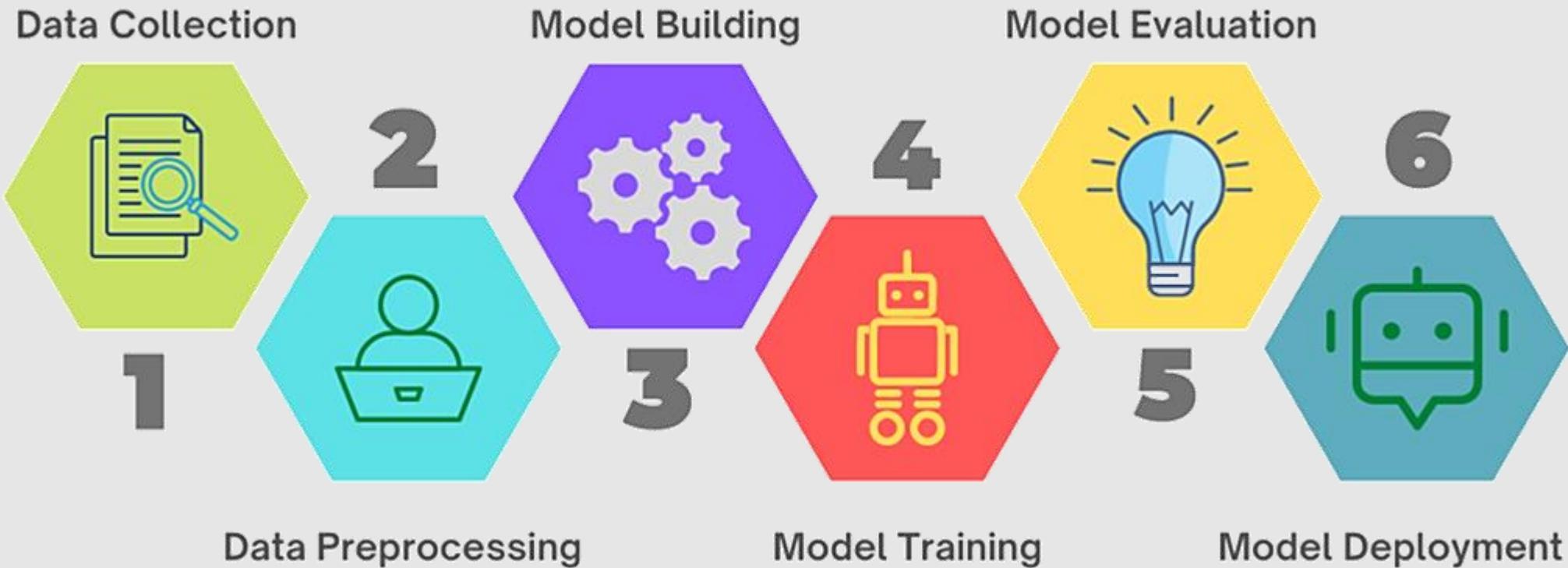
Different machine learning algorithms that's best fit for the above case will be tried out..

Given Dataset

S.No	Variables	Units	Description
1	Cement	kg in a m ³ mixture	Feature/Input
2	Blast FurnaceSlag	kg in a m ³ mixture	Feature/Input
3	Fly Ash	kg in a m ³ mixture	Feature/Input
4	Water	kg in a m ³ mixture	Feature/Input
5	Superplasticizer	kg in a m ³ mixture	Feature/Input
6	Coarse Aggregate	kg in a m ³ mixture	Feature/Input
7	Fine Aggregate	kg in a m ³ mixture	Feature/Input
8	Age	kg in a m ³ mixture	Feature/Input
9	Concrete compressive strength	kg in a m ³ mixture	Target/Output

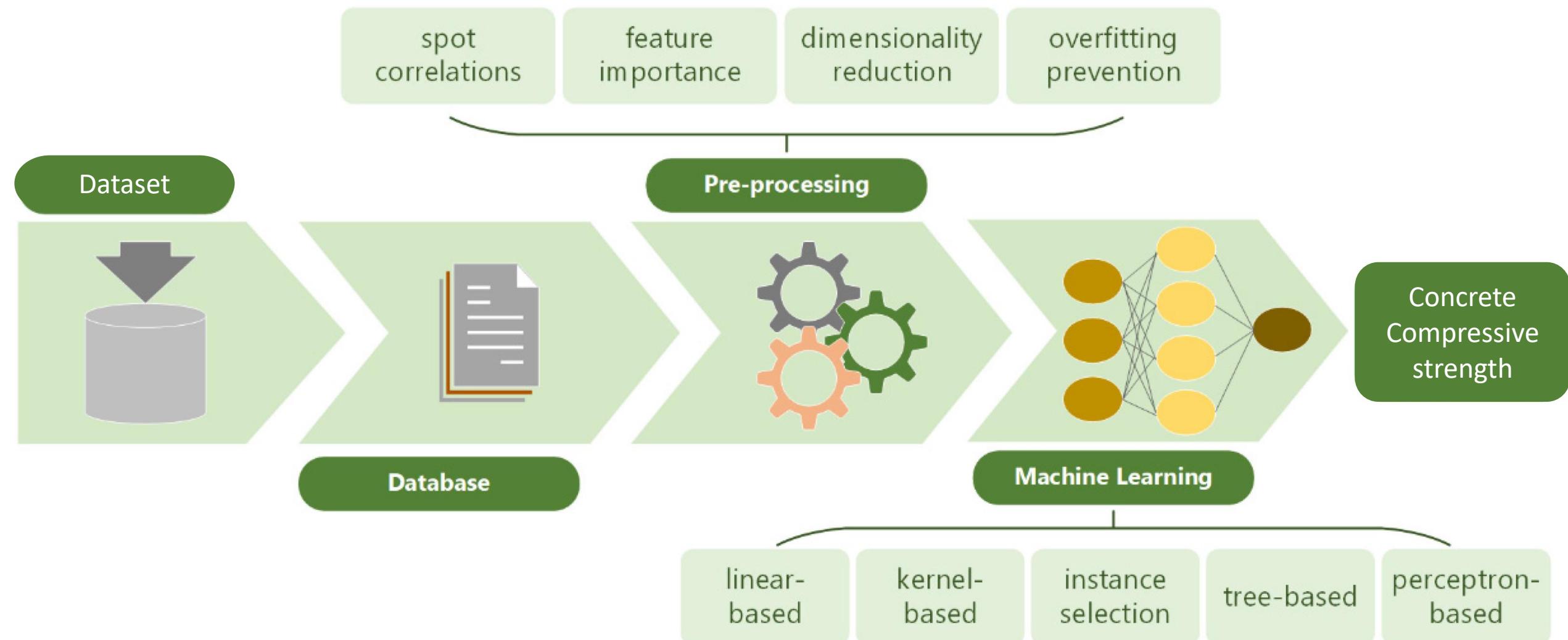


Steps



The above mentioned process steps were followed for the project

Steps



The above mentioned process steps were followed for the project

Import Packages and Create Dataframe

```
#IMPORTING THE NECESSARY LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import warnings
import sys

import missingno as mno

import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

```
# loading the dataset
dataset = pd.read_csv("D:/DATA/iNeuron/Concrete/concrete_data.csv")

target = dataset['concrete_compressive_strength']
features = dataset.iloc[:-1]

original_dataset = dataset.copy
display(dataset.head())
```

The above mentioned primary libraries and packages were imported and used for the project

Dataset loaded and Created Dataframe

	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
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

INFORMATION :
The Datset consists of
Features = 9
Total Samples = 1030

```
# loading the dataset
dataset = pd.read_csv("D:/DATA/iNeuron/Concrete/concrete_data.csv")

target = dataset['concrete_compressive_strength']
features = dataset.iloc[:-1]

original_dataset = dataset.copy
display(dataset.head())
```

The given Concrete dataset is loaded into the python environment

Checking the Dataframe

```
dataset.describe().T
```

		count	mean	std	min	25%	50%	75%	max
	cement	1030.0	281.167864	104.506364	102.00	192.375	272.900	350.000	540.0
	blast_furnace_slag	1030.0	73.895825	86.279342	0.00	0.000	22.000	142.950	359.4
	fly_ash	1030.0	54.188350	63.997004	0.00	0.000	0.000	118.300	200.1
	water	1030.0	181.567282	21.354219	121.80	164.900	185.000	192.000	247.0
	superplasticizer	1030.0	6.204660	5.973841	0.00	0.000	6.400	10.200	32.2
	coarse_aggregate	1030.0	972.918932	77.753954	801.00	932.000	968.000	1029.400	1145.0
	fine_aggregate	1030.0	773.580485	80.175980	594.00	730.950	779.500	824.000	992.6
	age	1030.0	45.662136	63.169912	1.00	7.000	28.000	56.000	365.0
	concrete_compressive_strength	1030.0	35.817961	16.705742	2.33	23.710	34.445	46.135	82.6

The descriptive statistics of the dataset used for the project was checked

Checking the Dataframe

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   cement           1030 non-null   float64 
 1   blast_furnace_slag 1030 non-null   float64 
 2   fly_ash          1030 non-null   float64 
 3   water            1030 non-null   float64 
 4   superplasticizer 1030 non-null   float64 
 5   coarse_aggregate 1030 non-null   float64 
 6   fine_aggregate   1030 non-null   float64 
 7   age              1030 non-null   int64  
 8   concrete_compressive_strength 1030 non-null   float64 
dtypes: float64(8), int64(1)
memory usage: 72.6 KB
```

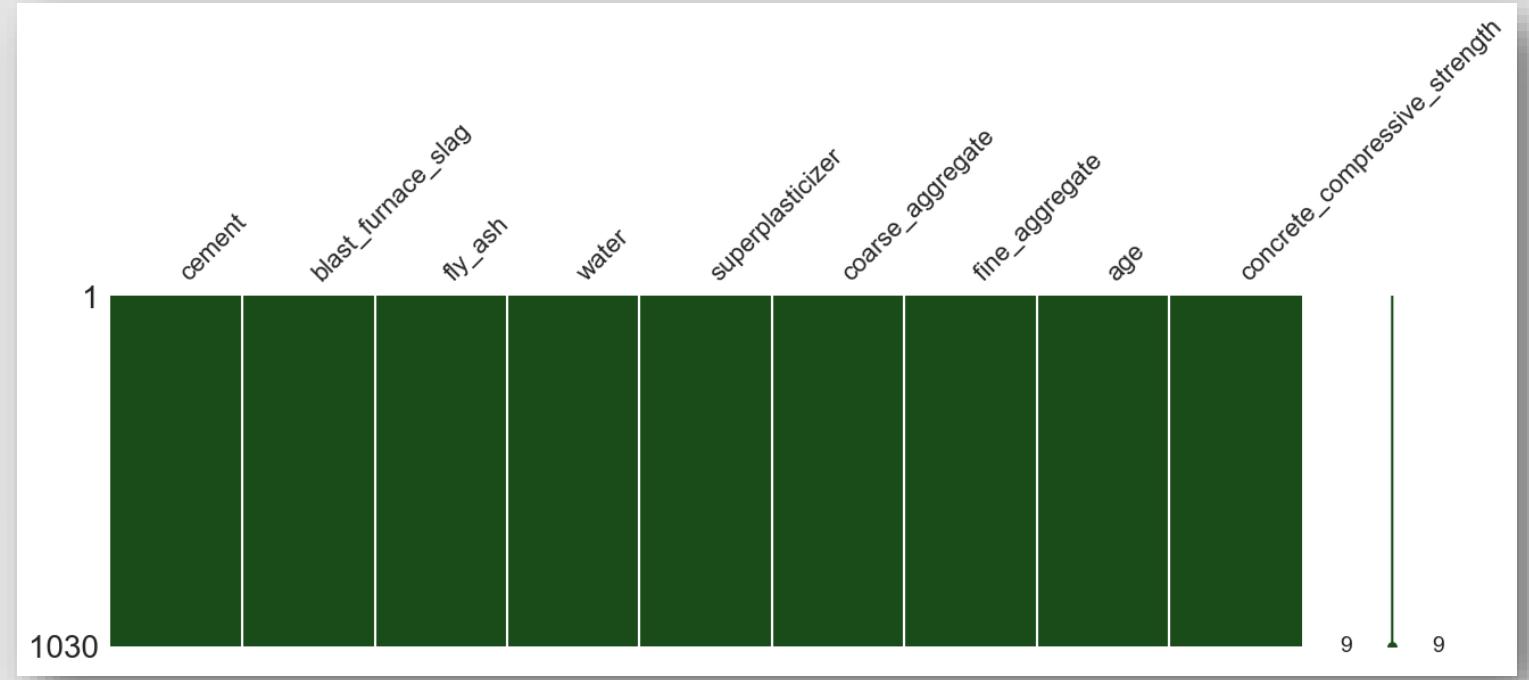
The records of the Dataframe was checked

- 1) In the given dataset, All the columns are numerical
- 2) All the columns datatype belong to float64,
Except for the column 'age'
- 3) The data has 8 quantitative input variables and
only one quantitative output variable - concrete_compressive_strength

Checking the Dataframe

```
# Checking for missing values  
dataset.isnull().sum()
```

```
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
```



The records of the Dataframe was checked – 9 columns were found with non null values

Checking the Dataframe

```
# Checking the dataset for duplicates  
  
print("The number of duplicated rows = ", dataset.duplicated().sum())
```

The number of duplicated rows = 25

- 1) In the given dataset, 25 duplicate rows were found
- 2) All the duplicate rows were dropped from the dataset
- 3) Initially the dataset had 1030 rows and now after dropping the dataset has now has 1005 rows.

```
# Delete duplicate rows  
dataset.drop_duplicates(inplace=True)
```

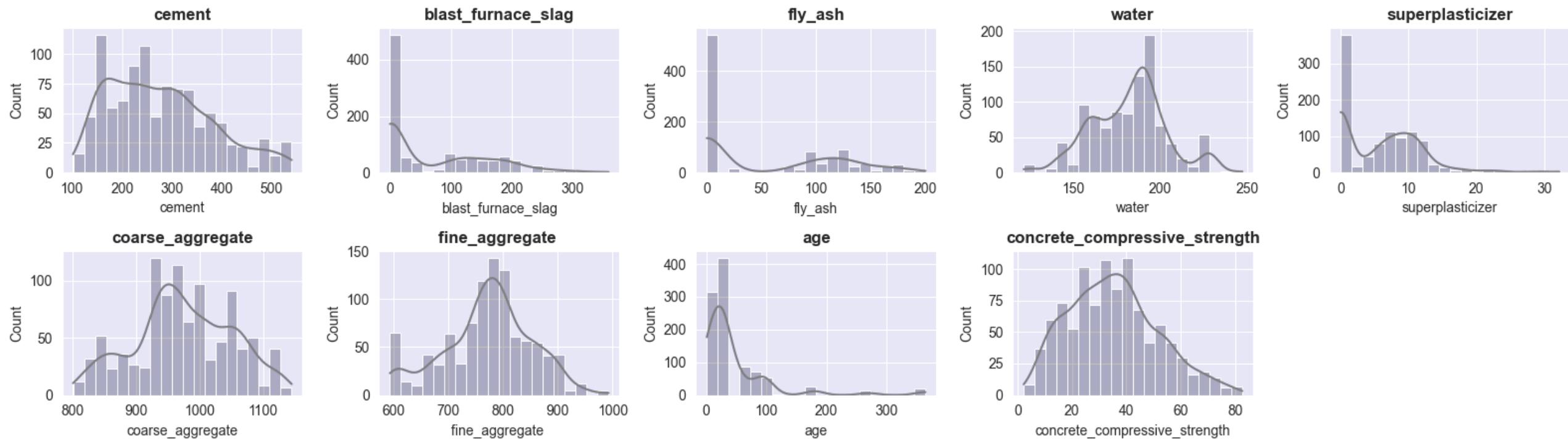
```
print('\n\033[1mINFORMATION : \033[0m \nAfter dropping the duplicates,
```

INFORMATION :
After dropping the duplicates, the Dataset contains now
Features = 9
Total Samples = 1005

Visualizing the Dataframe

The histogram plots of all the variables plotted

Histogram Plot



The distribution plots of all featured variables plotted to understand the distribution

Visualizing the Dataframe

Univariate analysis:

Cement - Right skewed distribution -- cement is skewed to higher values

Burnt Furnace Slag - Right skewed distribution -- slag is skewed to higher values and there are two gaussians

fly Ash - Right skewed distribution -- ash is skewed to higher values and there are two gaussians

Water - Moderately left skewed distribution

Superplasticizer - Right skewed distribution -- superplastic is skewed to higher values and there are two gaussians

Coarse aggregate - Moderately left skewed distribution

Fine aggregate - Moderately left skewed distribution

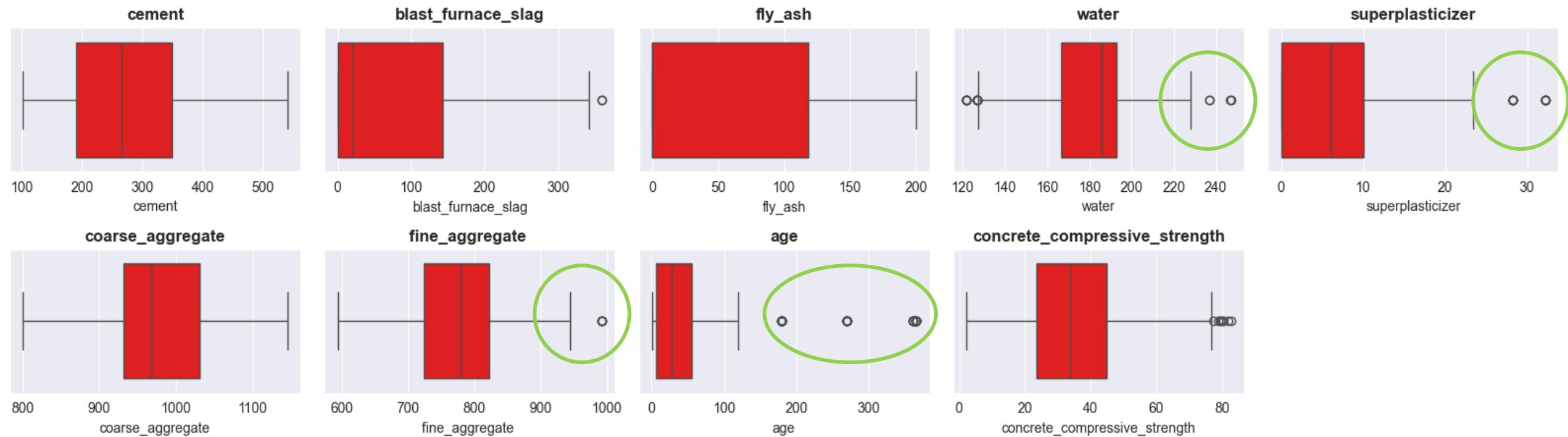
Age - Right skewed distribution -- age is skewed to higher values and there are five gaussians

Strength seems to be uniformly distributed

Visualizing the Dataframe

Detecting the Outliers

Box Plot

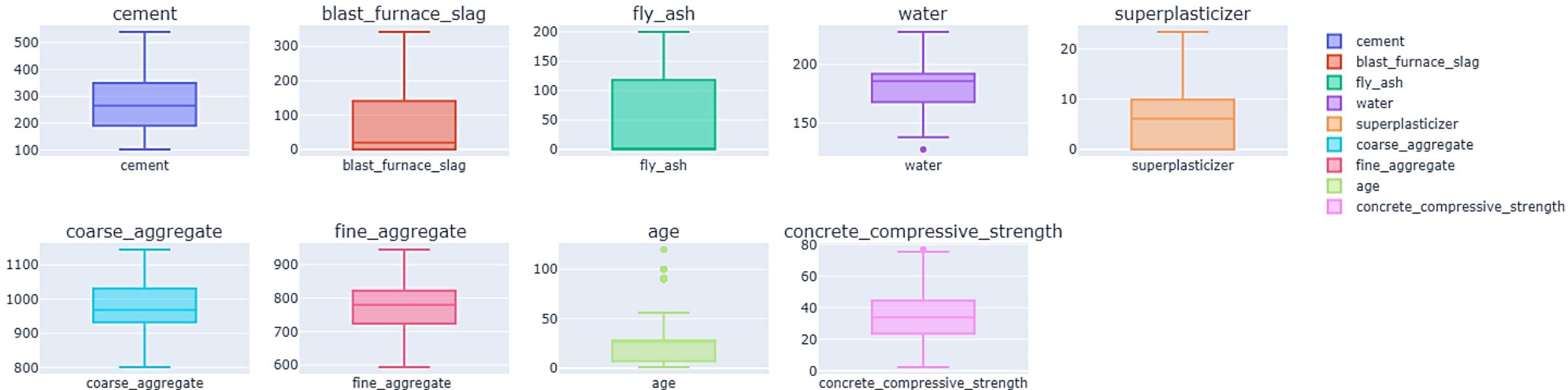


Several outliers have been detected , Next step is to remove outliers

Visualizing the Dataframe

Treating the Outliers

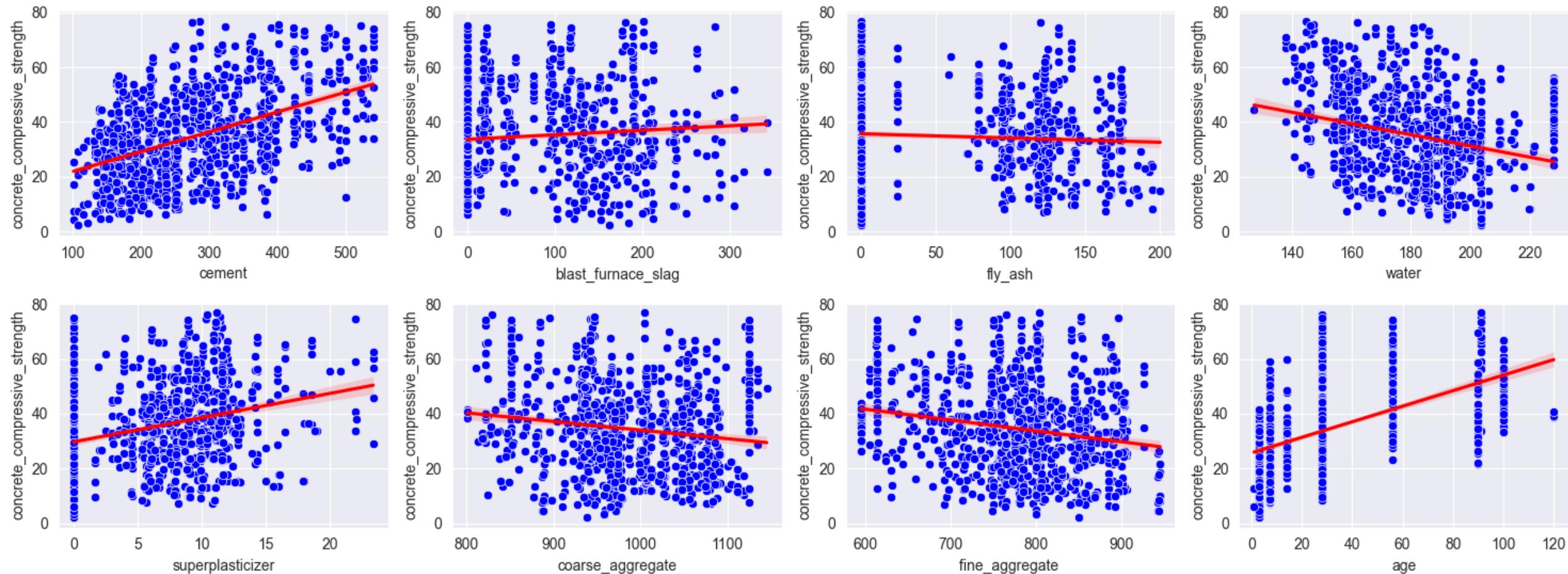
Box plots after treating outliers



Removing outliers and plotting the data to check if removed

Visualizing the Dataframe

Implots for Features vs Concrete Compressive Strength



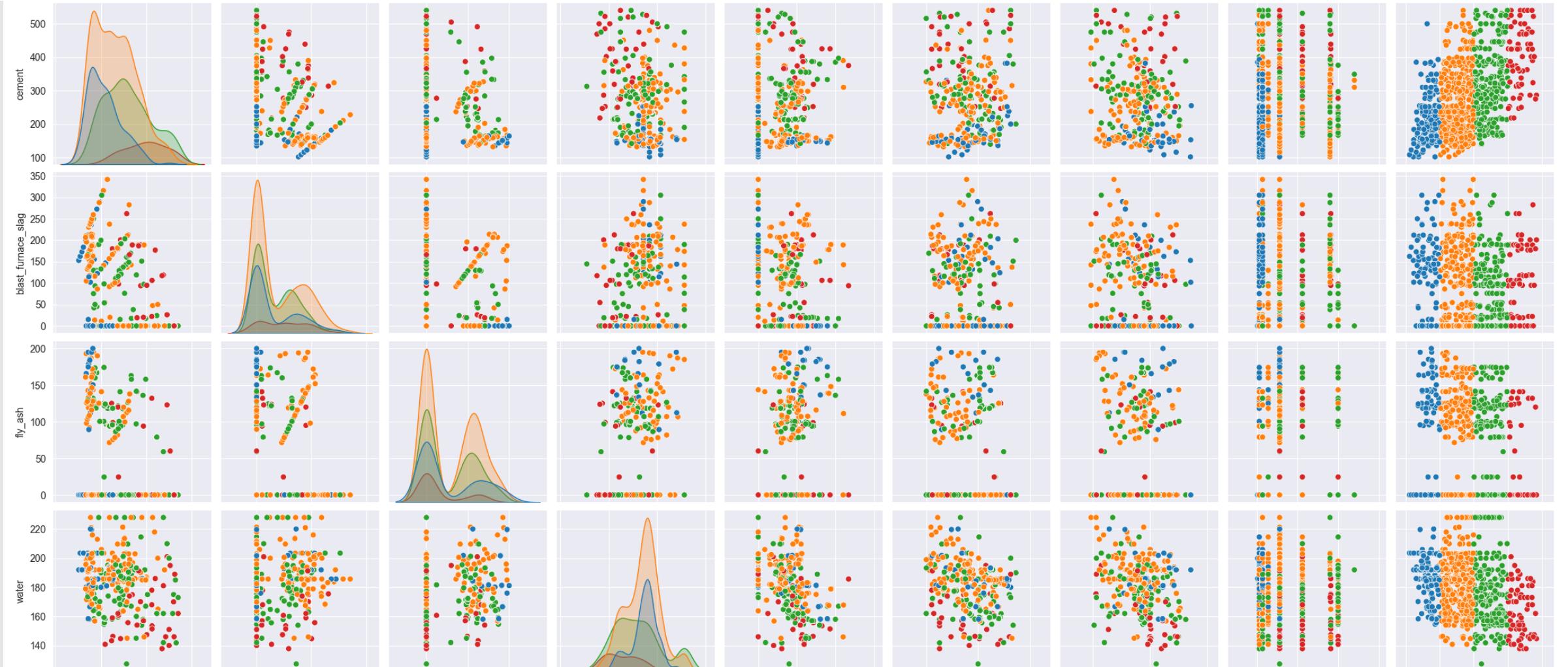
Visualizing the data to understand each variable using bivariate analysis

Visualizing the Dataframe

Bivariate analysis:

- 1) Cement : Widely spread data can be observed , with a linear increase for most of the other features.
- 2) Burnt-furnace-slag : Seems to have a large amount of right skewed and clustered below 200 kg m³ for most of the other features.
- 3) Fly Ash: Seems to have a dual distribution with values either zero or more than 100 kg m³ for most of the other features.
- 4) Water : Widely spread data can be observed. with a negative linear relation with most of the features.
- 5) Superplastic : Largely clustered around 5 to 15 units , with a linear increase for most of the other features.
- 6) Coarse aggregate : Widely spread data can be observed. with a negative linear relation with most of the features.
- 7) Fine aggregate : Widely spread data can be observed. with a negative linear relation with most of the features.
- 8) Age : A linear increase for most of the other features. Large data seen for a low age count can be seen in the given dataset.

Visualizing the Dataframe

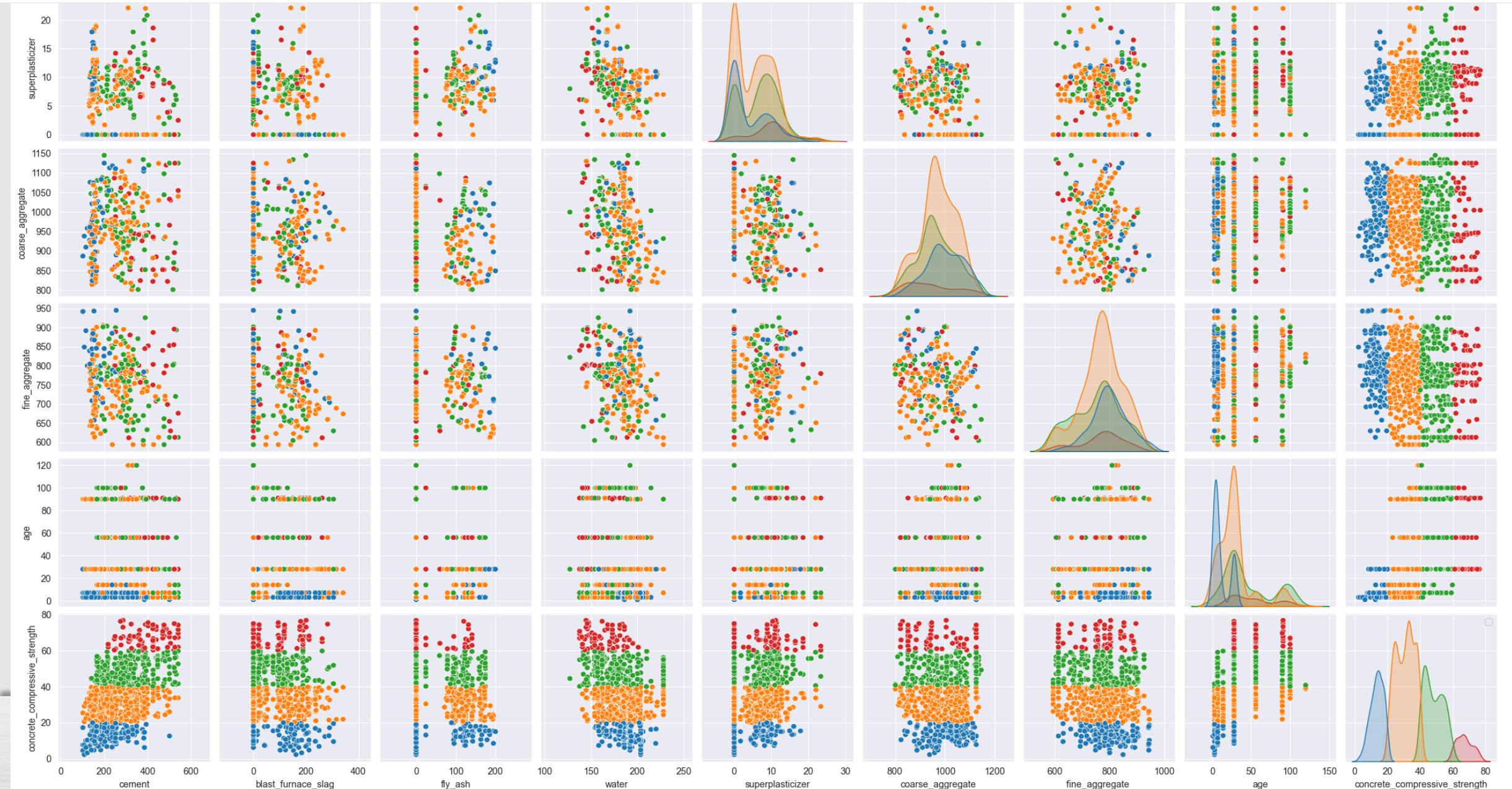


INFERENCE

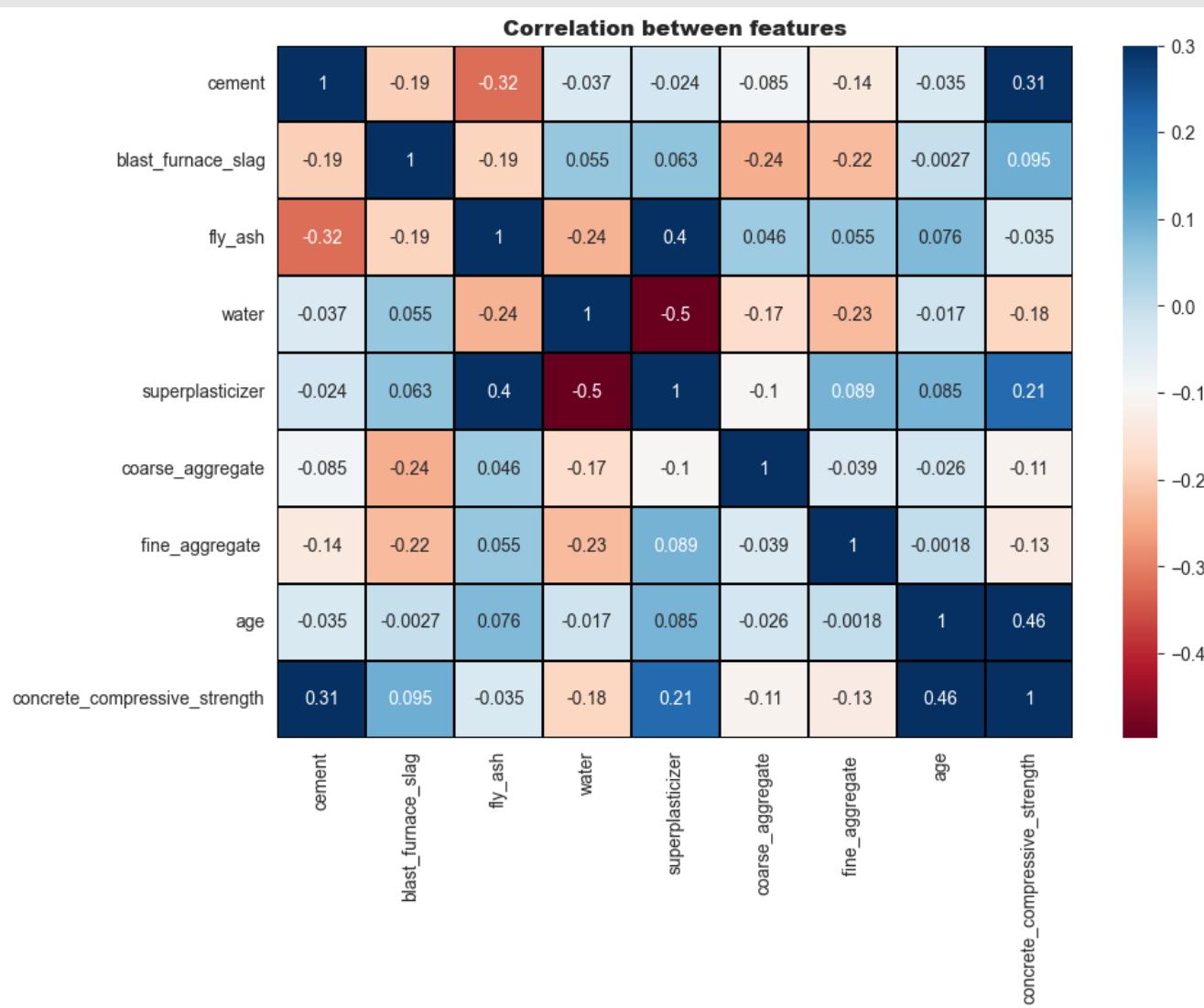
From the pair plots, a Right skewed distribution is observed for majority of the features. Except for the aggregates and water. At least 2 Gaussian (2 peaks) in furnace_Slag, fly_Ash, Superplasticer and Age, even though it's not unsupervised learning but in this dataset there are at least 2 clusters and there may be more.

Majority of the dataset is seen in the Range between 20 to 40 mPa for the concrete strength.

Visualizing the Dataframe



Visualizing the Dataframe



Concrete Compressive Strength

Cement, Superplasticizer and Age can be observed to have good linear relationship compared to other features

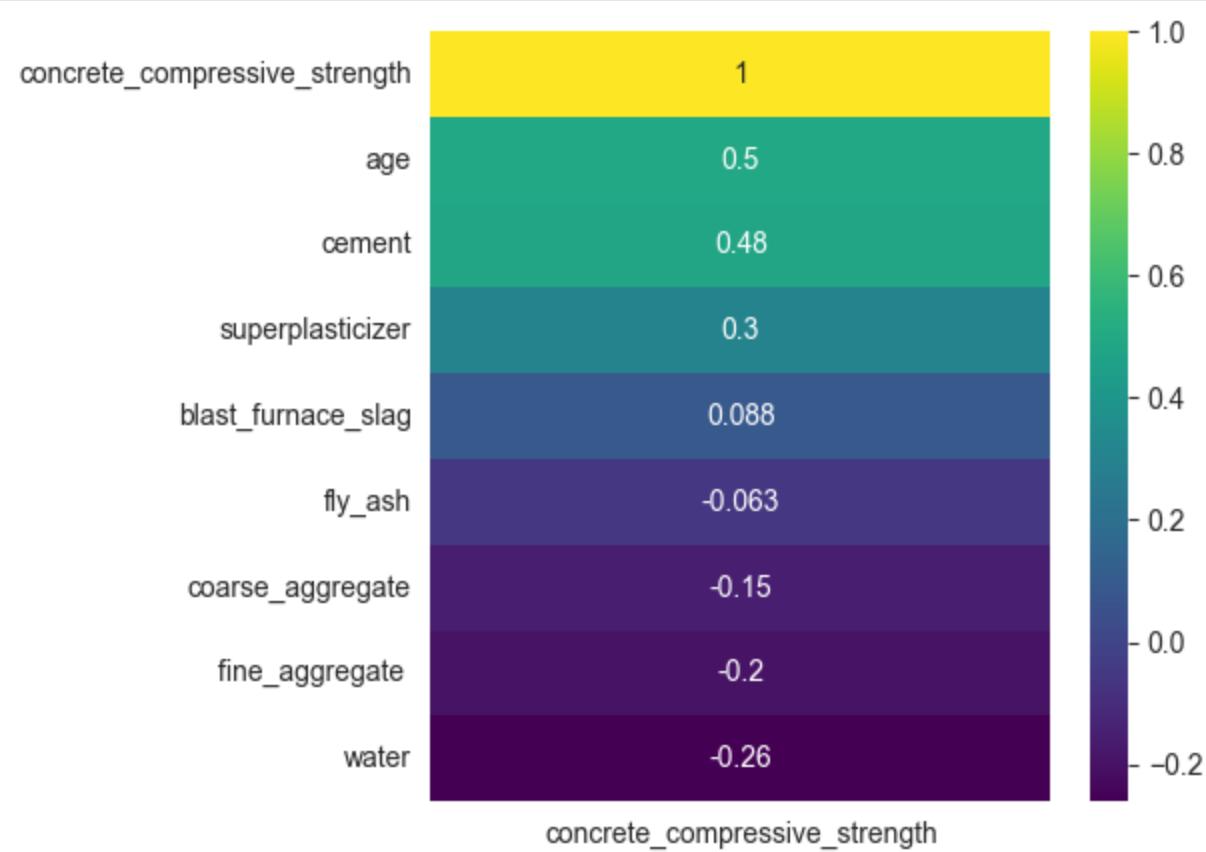
Inference –

Only few variables seem to correlate with Concrete strength

- 1) Cement
- 2) Age
- 3) Superplasticizer
- 4) Blast furnace slag

Visualizing the data using heatmap to understand correlation

Visualizing the Dataframe



Concrete Compressive Strength

Age ,Cement, Superplasticizer and slag can be observed to have good linear relationship compared to other features

Water has a total negative correlation

Model Building

```
# splitting into independant and target features

features = dataset.iloc[:, :-1]
target = dataset['concrete_compressive_strength']# splitting into independant and target features

features = dataset.iloc[:, :-1]
target = dataset['concrete_compressive_strength']
```

```
### Train test split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features_scaled,target, test_size= 0.3, random_state=40)

print(f'Train dataset shape: {X_train.shape}, {y_train.shape}')
print(f'Test dataset shape: {X_test.shape}, {y_test.shape}')
```

```
Train dataset shape: (703, 8), (703,)
Test dataset shape: (302, 8), (302,)
```

Model Building

```
### Train test split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(features_scaled,target, test_size= 0.3, random_state=40)

print(f'Train dataset shape: {x_train.shape}, {y_train.shape}')
print(f'Test dataset shape: {x_test.shape}, {y_test.shape}')

Train dataset shape: (703, 8), (703,)
Test dataset shape: (302, 8), (302,)
```

Performing Train - Test split on input data (To train and test model without Cross Validation and Hyper Parameter Tuning)

Data Modeling:

Since this is a Regression problem, I tried to build using few common Regression models for our training data and then compare performance of each model on test data to accurately predict target variable

Linear Regression

Decision Tree

Gradient Boosting Regr

Random Forest Regr

KNN Regressor

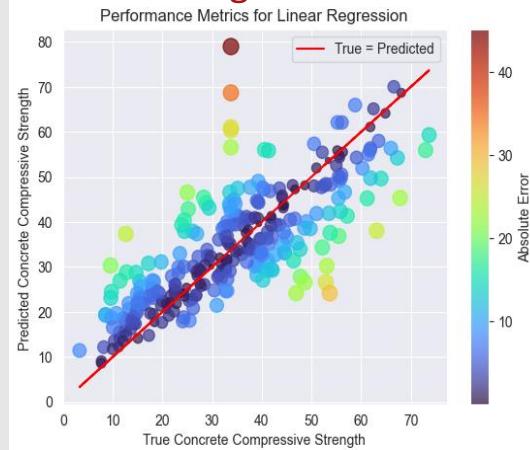
SVM Regressor

Ridge Regression

Lasso Regression

Models testing

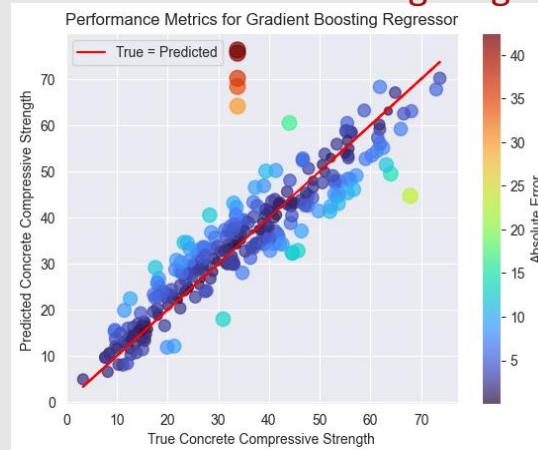
Linear Regression



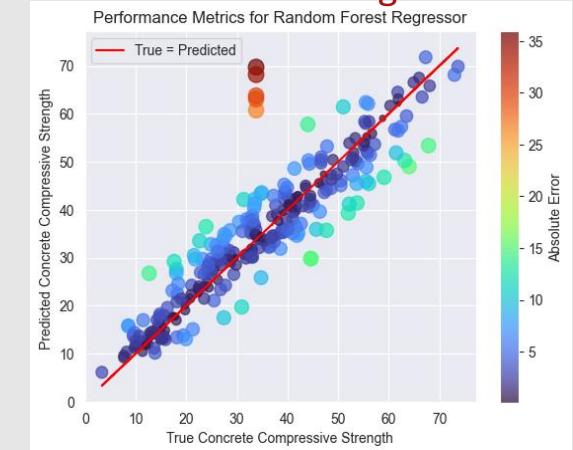
Decision Tree



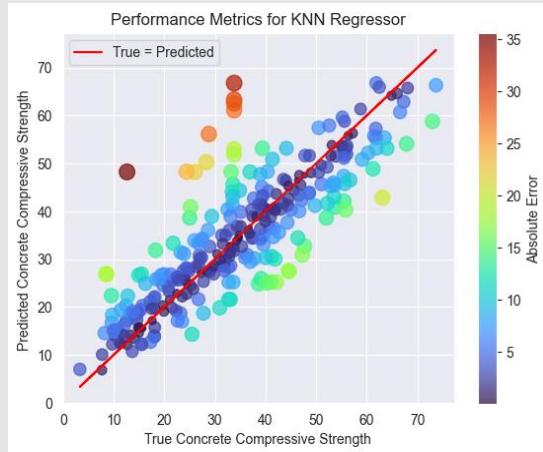
Gradient Boosting Regr



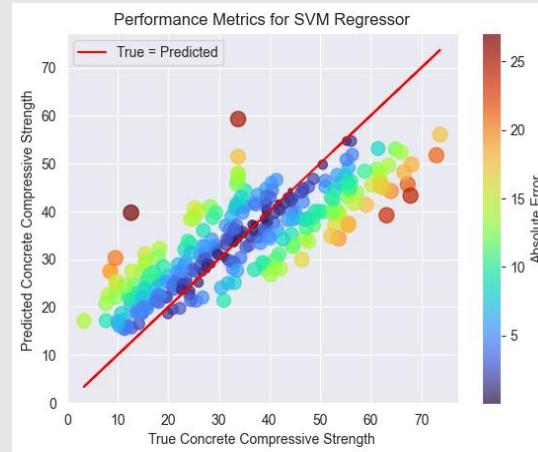
Random Forest Regr



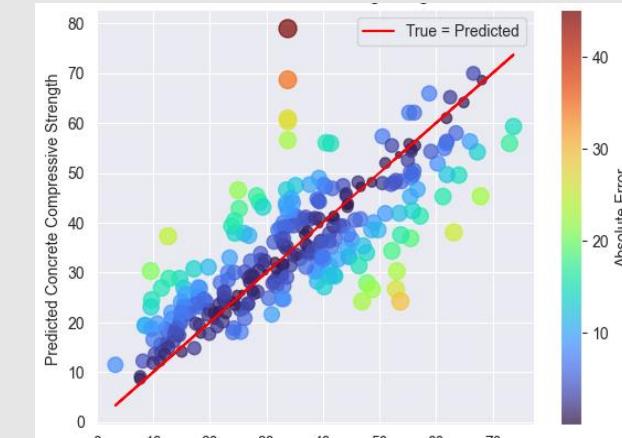
KNN Regressor



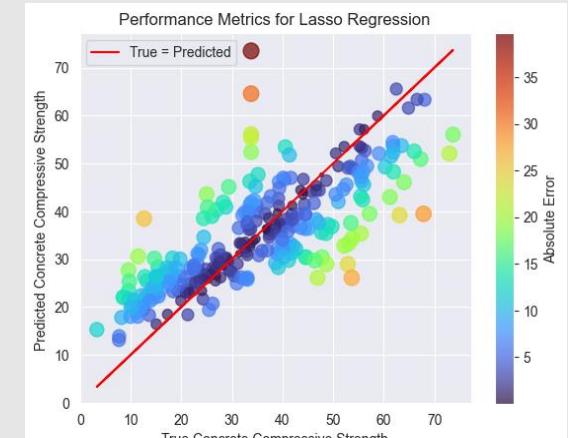
SVM Regressor



Ridge Regression



Lasso Regression



Performing Train - Test split on input data (To train and test model without Cross Validation and Hyper Parameter Tuning)

Models testing Output

Comparing all the models – to find out the best model

S No.	Model No.	RMSE	MAE	MSE	R2
1	Linear Regression	9.390438	6.860813	88.180334	0.622271
2	Decision Tree	8.627499	5.691987	74.433745	0.681156
3	Gradient Boosting Regr	7.039062	4.541052	49.548396	0.787755
4	Random Forest Regr	6.443462	4.290236	41.518202	0.822153
5	KNN Regressor	8.303886	5.885437	68.954524	0.704627
6	SVM Regressor	9.124020	7.199210	83.247745	0.643400
7	Ridge Regression	9.387114	6.862371	88.117918	0.622538
8	Lasso Regression	9.606232	7.417111	92.279699	0.604711

Inference

Looking at the data we can clearly see that the Random Forest algorithm does a better job in all the metrics as compared to all the other classifiers.

Models testing Output



Inference
Looking at the data we can clearly see that the Random Forest algorithm does a better job in all the metrics as compared to all the other classifiers.

Models testing Output

Inference

Random Forest algorithm was chosen to do the final prediction as it gave the best parameters.

```
n_estimators = [100, 200, 500]  
max_depths = [30, 50, 70]  
min_samples_leafs = [2, 5, 10]
```

With the above mentioned parameters Model hypertuning was carried out for random forest algorithm

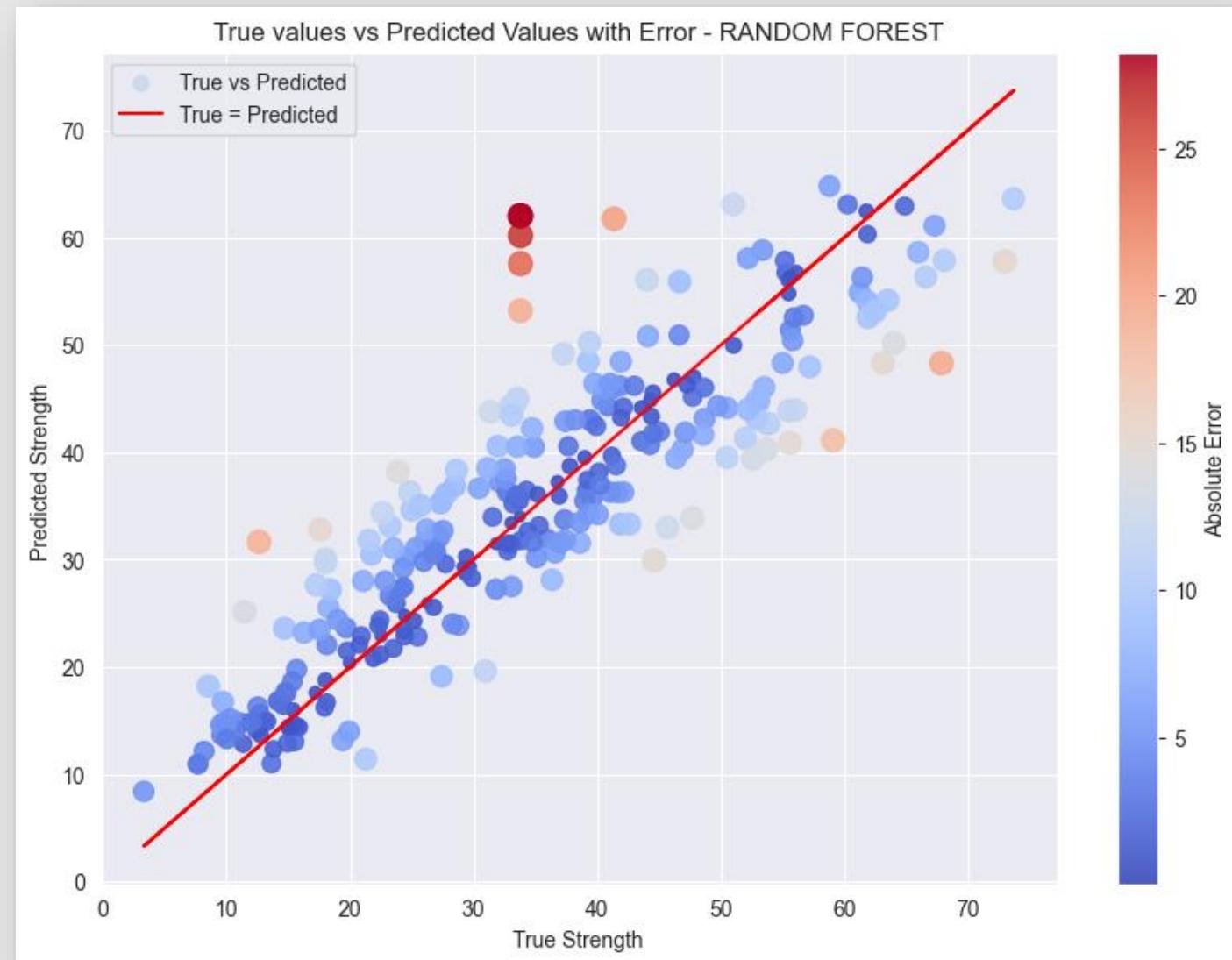
An improved R2 of 83.13 % was obtained

Model No.	RMSE	MAE	MSE	R2		
Random Forest Regr	6.443462	4.290236	41.518202	0.822153		
Estimator	Max Depth	Min Samples Leaf	R2 Score	MSE	MAE	RMSE
200	30	2	0.831345	39.37236	4.219989	6.27474

Final Model Output

	True Tc	Predicted Tc	Absolute error
767	31.35	43.793848	12.443848
518	23.25	33.131331	9.881331
978	28.99	28.928422	0.061578
929	21.91	20.799060	1.110940
71	28.80	23.894828	4.905172
...
249	13.82	12.351441	1.468559
874	36.80	37.240048	0.440048
579	22.63	34.375902	11.745902
624	27.53	32.713666	5.183666
751	52.61	39.420591	13.189409

302 rows × 3 columns



Final Model Deployment

Concrete Compressive Strength Predictor

Cement :
blast_furnace_slag :
fly_ash :
water :
superplasticizer :
coarse_aggregate :
fine_aggregate :
age No of Ds

CONCRETE COMPRESSIVE STRENGTH PREDICTION IS = mPa

Concrete Compressive Strength Predictor

Cement :
blast_furnace_slag :
fly_ash :
water :
superplasticizer :
coarse_aggregate :
fine_aggregate :
age

CONCRETE COMPRESSIVE STRENGTH PREDICTION IS = mPa

- Project By Paul Praveen

Concrete Compressive Strength Predictor

Cement :
blast_furnace_slag :
fly_ash :
water :
superplasticizer :
coarse_aggregate :
fine_aggregate :
age No of Ds

CONCRETE COMPRESSIVE STRENGTH PREDICTION IS = 66.38167557642474 mPa

- Project By Paul Praveen

Final Model Deployment

The image shows two side-by-side instances of Microsoft Visual Studio Code (VS Code) running on a Windows operating system. Both instances have the title bar "Concrete" and are displaying the same code editor window.

Code Editor Content:

```
File Edit Selection View Go Run Terminal Help < > Concrete
EXPLORER requirements.txt app.py python.config
CONCRETE .ebextensions python.config
artifacts catboost_info Concrete.egg-info logs
notebook data
1. EDA concrete-compre...
2. MODEL TRAINING C...
Concrete Compressive S...
src templates venv .gitignore
app.py application.py README.md requirements.txt setup.py

app.py > predict_datapoint
1 from flask import Flask,request,render_template
2 import numpy as np
3 import pandas as pd
4 import os
5
6 from sklearn.preprocessing import StandardScaler
7 from src.pipeline.predict_pipeline import CustomData,PredictPipeline
8
9 application=Flask(__name__)
10
11 app=application
12
13 ## Route for a home page
14
15 @app.route('/')
16 def index():
17     return render_template('index.html')
18
19 @app.route('/predictdata',methods=['GET','POST'])
20 def predict_datapoint():
21     if request.method=='GET':
22         return render_template('home.html')

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

Terminal Output (Left Window):

```
(d:\DATA\iNeuron\MLproject\Concrete\venv) D:\DATA\iNeuron\MLproject\Concrete>python app.py
* Serving Flask app 'app'
* Debug mode: off
```

Terminal Output (Right Window):

```
(d:\DATA\iNeuron\MLproject\Concrete\venv) D:\DATA\iNeuron\MLproject\Concrete>python app.py
* Serving Flask app 'app'
* Debug mode: off
cement blast_furnace_slag fly_ash water superplasticizer coarse_aggregate fine_aggregate age
0 540.0 142.0 5.0 228.0 3.0 1055.0 676.0 360.0
Before Prediction
Mid Prediction
Before Loading
After Loading
after Prediction
```

Git hub Repository

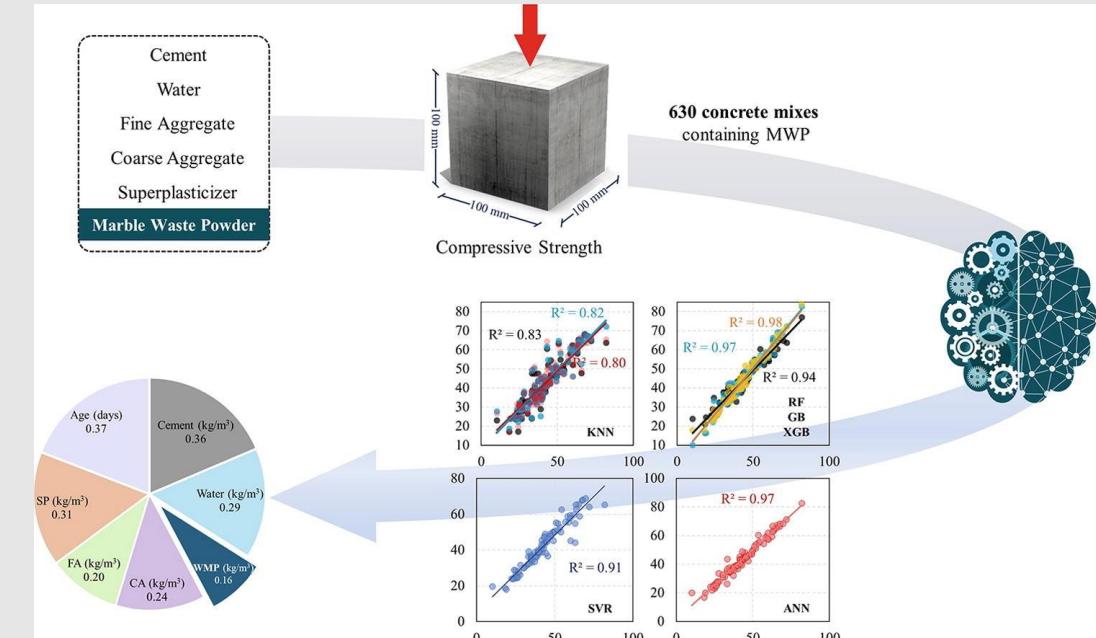
Concrete_Compressive_Strength_ML_iNeuron Public

PPraveen2230	Deployment and config	4ab790b · yesterday 7 Commits
.ebextensions	Deployment and config	yesterday
artifacts	Hyperparamter tuning	4 days ago
catboost_info	Hyperparamter tuning	4 days ago
notebook	Deployment and config	yesterday
src	Model Prediction Pipeline	4 days ago
templates	Deployment and config	yesterday
.gitignore	Create .gitignore	4 days ago
Procfile	Deployment and config	yesterday
README.md	first commit	4 days ago
app.py	Model Prediction Pipeline	4 days ago
application.py	Deployment and config	yesterday
requirements.txt	Deployment and config	yesterday
setup.py	Second commit	4 days ago

README

Problem Statement:

The Project being uploaded in my Git hub repository



https://github.com/PPraveen2230/Concrete_Compressive_Strength_ML_iNeuron.git

Conclusion

1. Concrete compressive strength using original features with an accuracy of 84 % on test data were predicted from this project
2. Results from various methods of analysis shows us that we got the best accuracy from original features and followed below steps to gain that much of accuracy.
 - a. As mentioned in Multi-variate analysis, there are some non-linear(curvy-linear) relationship within independent features as well as with target variable hence polynomial features were implemented.
 - b. Simple linear regression with polynomial features with degree = 2 performs better on both training and test set with 1% difference.
3. We had 25 duplicate instances in dataset and dropped those duplicates.
4. We had outliers in 'Water', 'Superplastic', 'Fineagg', 'Age' and 'Strength' column also, handled these outliers by replacing every outlier with upper and lower side of the whisker.
5. Except 'Cement', 'Superplastic', 'slag' and 'Age' features, all other features are having very weak relationship with concrete 'Strength' feature and does not account for making statistical decision (of correlation).
6. Range of clusters in this dataset is 2 to 6
7. No missing values were found in dataset.
8. **Finally Random Forest Regressor model with an accuracy of 84 % is our best model.**

Model No.	RMSE	MAE	MSE	R2
Random Forest Regr	6.443462	4.290236	41.5182	0.822153
Gradient Boosting Regr	7.039062	4.541052	49.5484	0.787755
KNN Regressor	8.303886	5.885437	68.95452	0.704627

Estimator	Max Depth	Min Samples Leaf	R2 Score	MSE	MAE	RMSE
200	30	2	0.831345	39.37236	4.219989	6.27474

