PROJECT

Analysis on Concrete Strength

TEAM-4

A.AKHILA

CH.RAHUL

R.MITALI

K.BHANU PRASAD

V.NANDINI

Bhavan's Vivekananda College | BSc Hons Data Science

OBJECTIVE

Concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate.
Our objective is to predict the strength of concrete based on varied composition levels of the ingredients

THE PATH

We followed numerous ML Algorithms to fit a predictive model to this data inorder to determine the strength of high-performance concrete

DATA INTRODUCTION :

The data consists of concrete strength information of 1030 data points. Some of The data variables include Cement content, Ash content, Slag content, Water content, Amount of super plasticizer, Age of content, Strength of the content.

The actual concrete compressive strength (MPa) for a given mixture under a specific age (days) was determined from laboratory.
Data is in raw form

THE DATA CONSISTS OF CONCRETE STRENGTH INFORMATION OF 1029 DATA POINTS. SOME OF THE DATA VARIABLES INCLUDE

1. CEMENT CONTENT
2. ASH CONTENT
3. SLAG CONTENT
4. WATER CONTENT
5. AMOUNT OF SUPERPLASTICISER USED
6. AGE OF CEMENT
7. STRENGTH OF THE CEMENT

ATTRIBUTE INFORMATION

- CEMENT-Cement is a finely milled mineral powder, usually grey in colour.
- BLAST FURNACE SLAG-Blast furnace slag is a nonmetallic coproduct produced in the process.
- FLY ASH-ash produced in small dark flecks by the burning of powdered coal or other materials and carried into the air.
- WATER- H2O. Hydrogen monoxide.
- SUPER PLASTICIZER-are additives used in making high strength concrete.
- COARSE AGGREGATION-the portion of the aggregate used in concrete that is larger than about $^{3}/_{16}$ inch.
- FINE AGGREGATION-the portion of the aggregate used in concrete that is larger than about $^{3}/_{16}$ inch.
- CONCRETE AGE-Concrete age is the time since the moment water is added to the cement
- CONCRETE COMPRESSIVE STRENGTH-the capacity of concrete to withstand loads before failure.

	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
...	...	...	...	...	...	...	...	...	...
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	108.6	192.7	6.1	892.4	780.0	28	23.70
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.77
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40

1030 rows × 9 columns

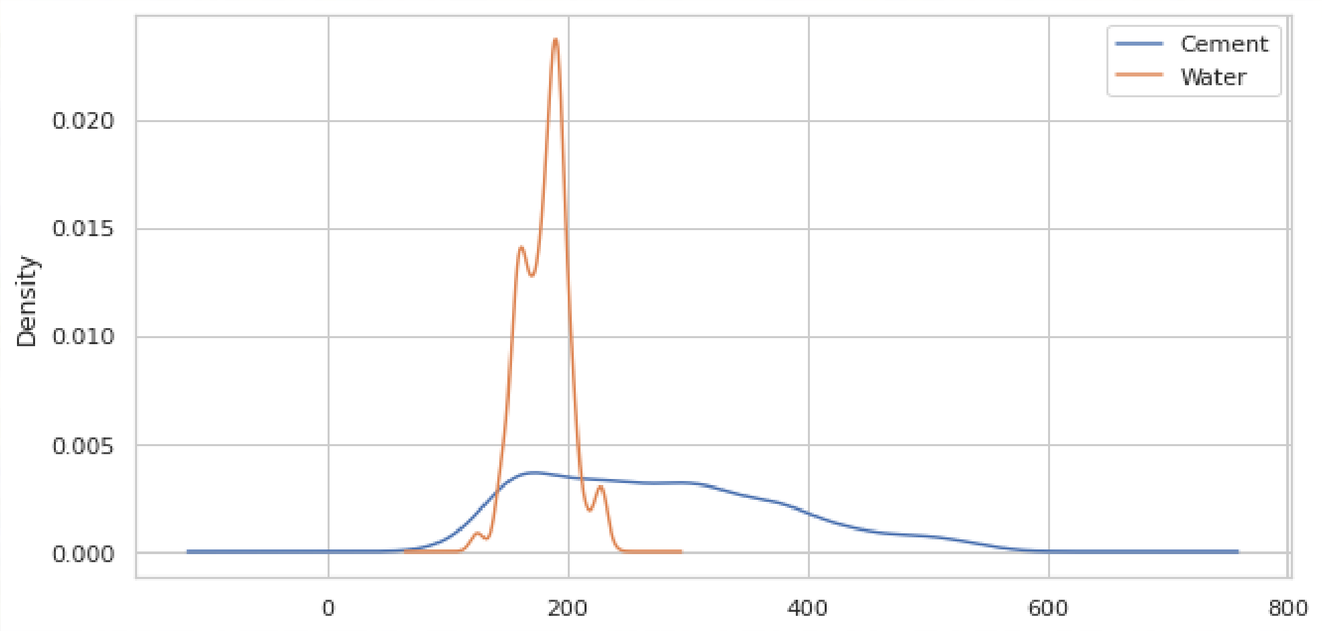
We have imported our data

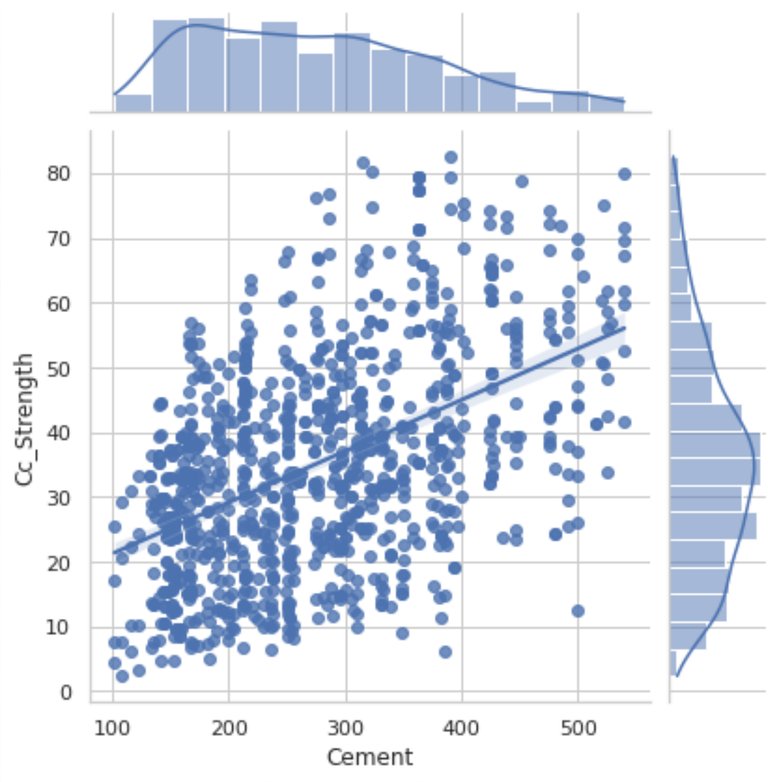
```
data.isnull().sum()
```

```
Cement              0
Furnace_Slag        0
Fly_Ash             0
Water               0
Super_Plasticizer   0
Coarse_agg          0
Fine_agg            0
Age                 0
Cc_Strength         0
dtype: int64
```

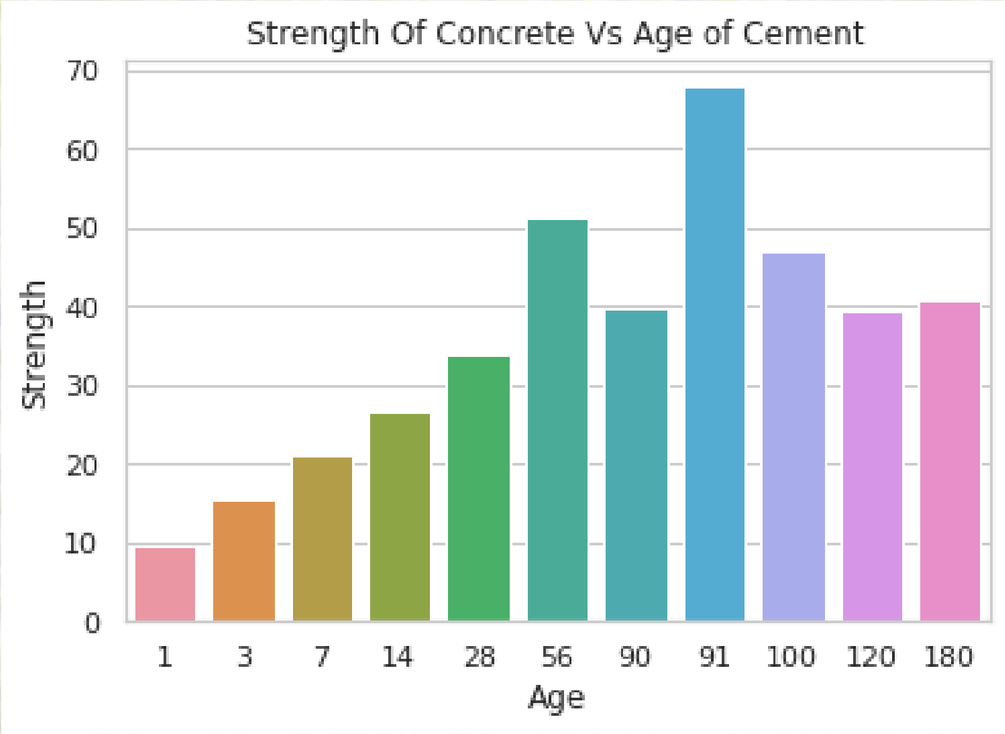
We didnt find any Missing or NULL values in our dataset.

EXPLORATORY DATA ANALYSIS

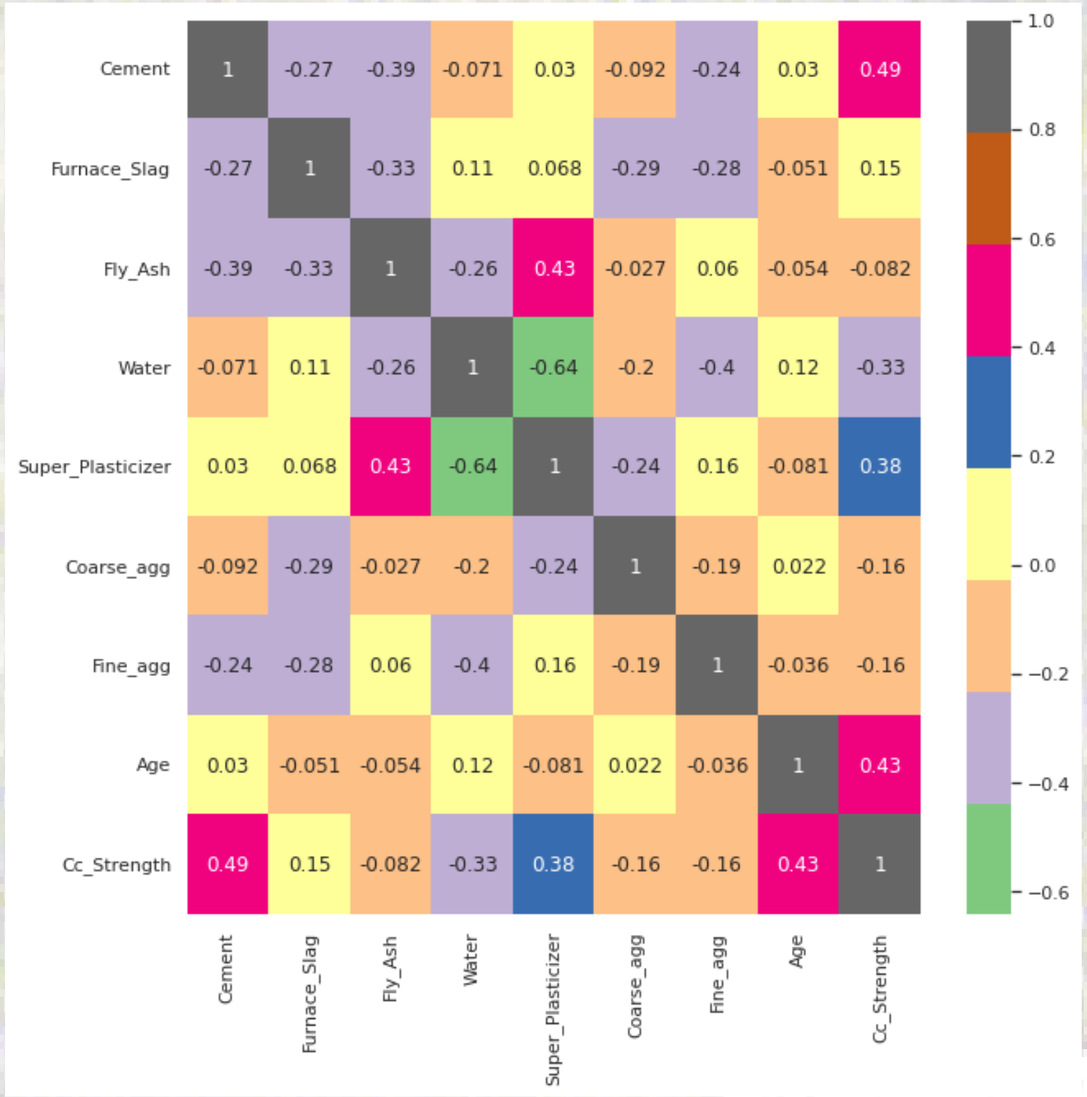




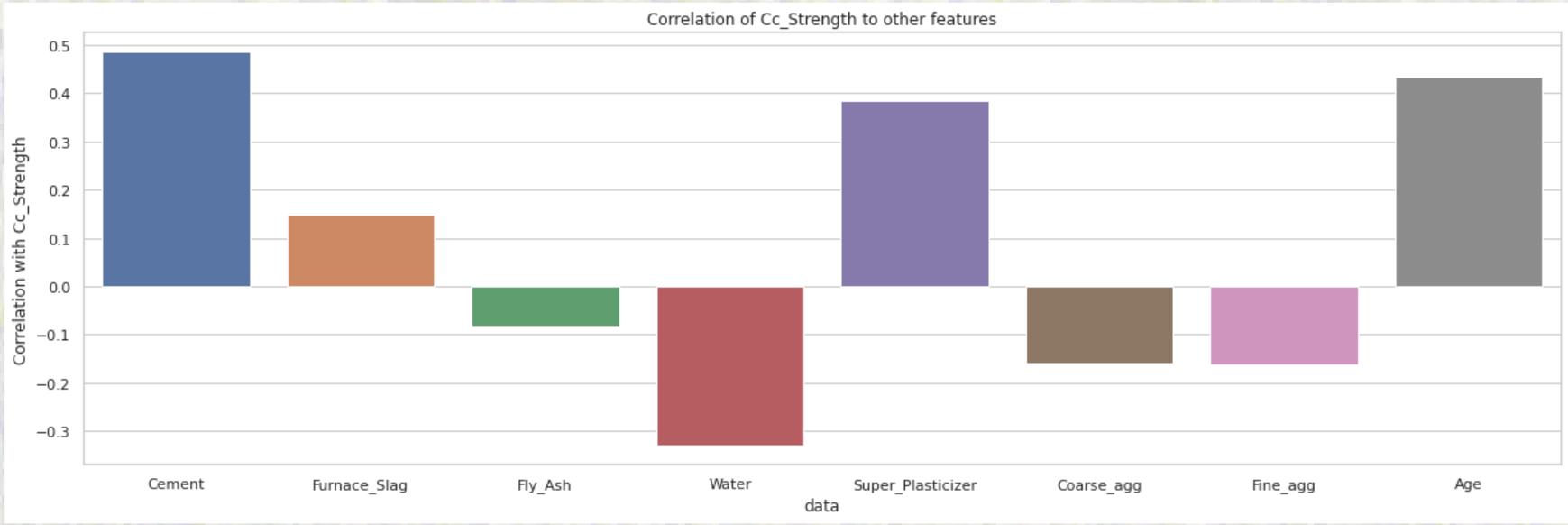
EXPLORATORY DATA ANALYSIS



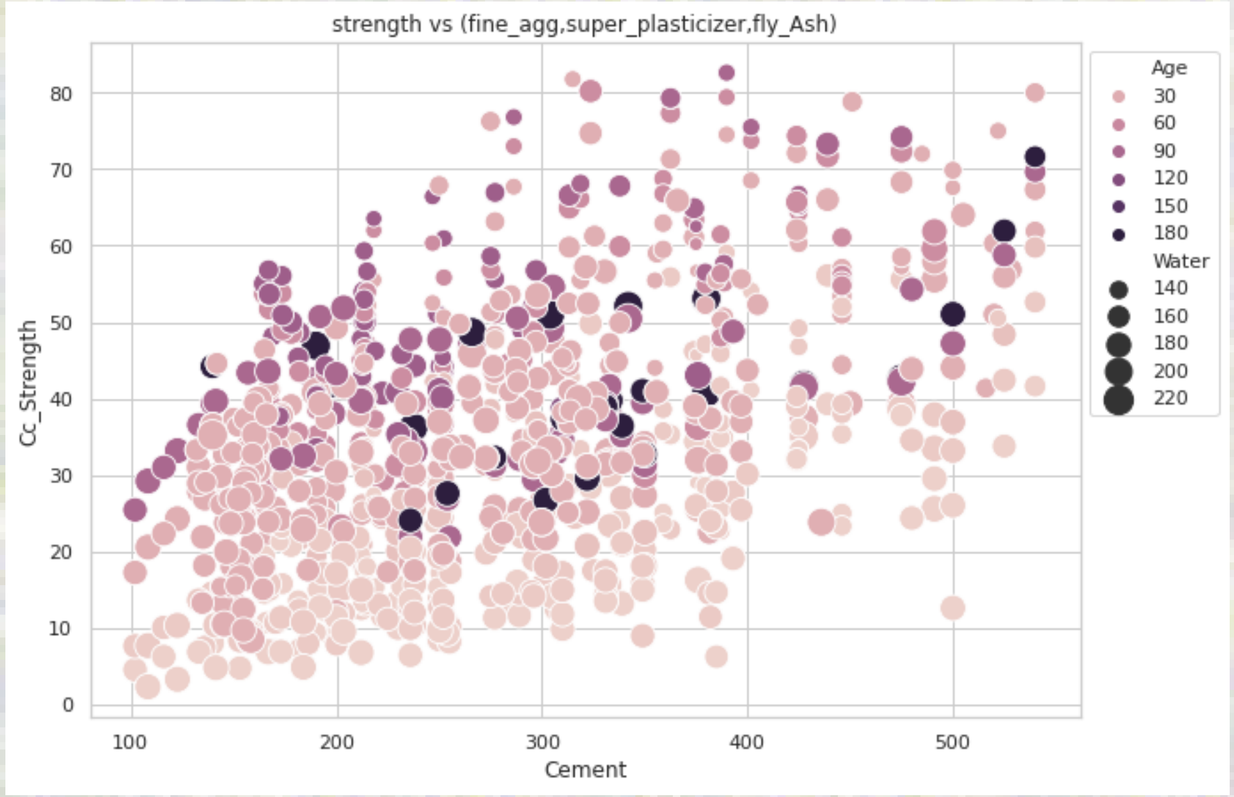
EXPLORATORY DATA ANALYSIS

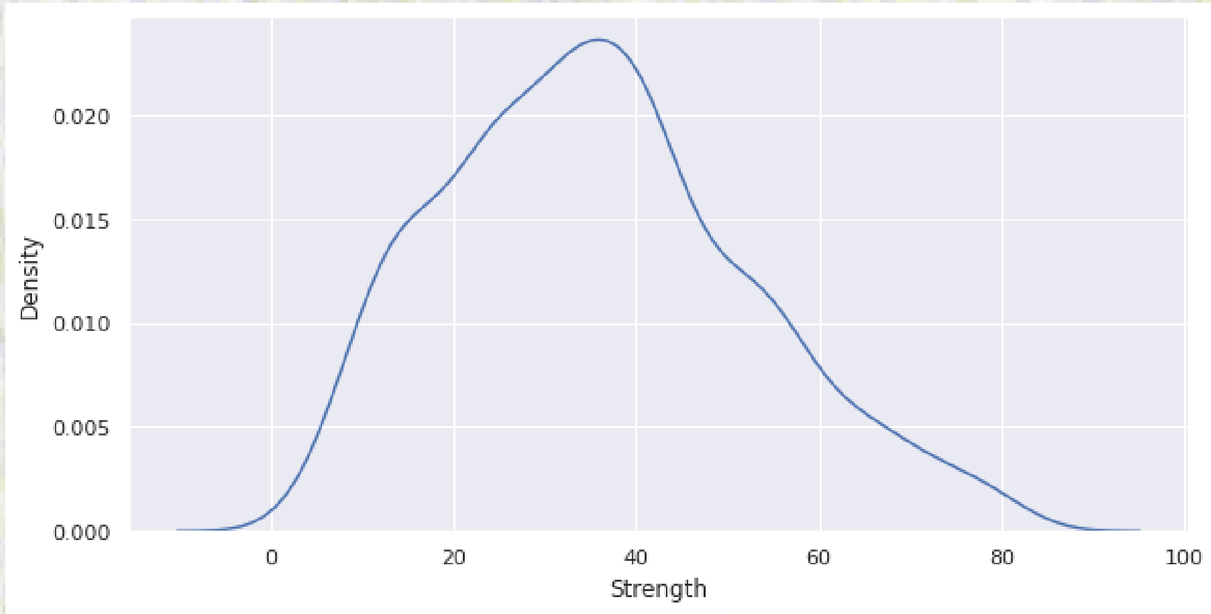


EXPLORATORY DATA ANALYSIS



EXPLORATORY DATA ANALYSIS





LINEAR REGRESSION

Train Test Split	Mae value
80:20	7.3842071
70:30	**6.872863071**
60:40	7.088558

NEURAL NETWORKS

These are the mean absolute error values when the train test ratio is taken as 80:20

ARCHITECTURE	EPOCHS	MAE
8-2-3-5-1	100	8.1872
8-8-8-5-1	150	8.1098
8-8-4-5-1	120	8.9919
8-6-4-2-1	170	7.9576
8-4-2-1	370	7.9454
8-3-5-6-1	70	7.9833

NEURAL NETWORKS

These are the mean absolute error values when the train test ratio is taken as 70:30

ARCHITECTURE	EPOCHS	MAE
8-6-4-2-1	100	8.8493
8-8-8-5-1	120	9.0958
8-7-2-5-1	90	9.4041
8-5-8-4-1	30	8.6334
8-3-5-6-1	35	8.6158
8-6-4-2-1	1000	8.3355

NEURAL NETWORKS

These are the mean absolute error values when the train test ratio is taken as 60:40

ARCHITECTURE	EPOCHS	MAE
8-3-4-7-1	55	8.7224
8-2-8-5-1	735	8.6123
8-3-5-6-1	70	8.4391
8-7-2-5-1	90	8.4344
8-3-5-6-1	35	8.9419

BAGGING

Train Test Ratio	Estimators	Mae value
80:20	**500**	**4.3228750661**
70:30	700	4.3273628632
60:40	750	4.7255579355

BOOSTING

GRADIENT BOOSTING		
80:20	**650**	**2.930060**
70:30	800	2.984306
60:40	750	3.207603

EXTREME	GRADIENT	BOOSTING
80:20	**500**	**2.84850**
70:30	750	3.05610
60:40	600	3.441065

ADAPTIVE BOOSTING		
80:20	650	6.425250
70:30	**350**	**6.352956**
60:40	750	6.594533

Algorithms	Mae values
GRADIENT BOOSTING	2.9300
ADABOOST	6.3529
XG BOOST	2.84850
LINEAR REGRESSION	6.87286
BAGGING	4.32287
NEURAL NETWORKS	7.9454

CONCLUSIONS

Strength of concrete had a very close relation with the amount of water and cement added

Selecting the cement of right age can significantly improve the strength of the cement

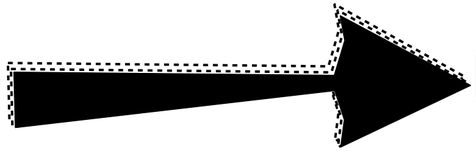
XGBM produced best Outcome

THANK YOU

CLICK HERE FOR THE CODE

APPENDIX

```
import pandas as pd
import seaborn as sns
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import numpy as np
from scipy import stats
import seaborn as sns
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf
# allow plots to appear directly in the notebook
%matplotlib inline
```



IMPORTING LIBRARIES

GETTING DATA

```
data = pd.read_csv("/content/Concrete_Data.csv")
data
```

```
data= data.rename(columns = {
    'Cement (component 1)(kg in a m^3 mixture)':'Cement',
    'Blast Furnace Slag (component 2)(kg in a m^3 mixture)':'Furnace_Slag',
    'Fly Ash (component 3)(kg in a m^3 mixture)':'Fly_Ash',
    'Water  (component 4)(kg in a m^3 mixture)':'Water',

    'Superplasticizer (component 5)(kg in a m^3 mixture)':'Super_Plasticizer',
    'Coarse Aggregate  (component 6)(kg in a m^3 mixture)':'Coarse_agg',
    'Fine Aggregate (component 7)(kg in a m^3 mixture)':'Fine_agg',
    'Age (day)':'Age',
    'Concrete compressive strength(MPa, megapascals) ': 'Cc_Strength'})

data
```

RENAMING DATA

```
[ ]  # independent variables
     x = data.drop(['Cc_Strength'],axis=1)
     # dependent variables
     y = data['Cc_Strength']
```

```
[ ]  # importing train_test_split
     from sklearn.model_selection import train_test_split
     xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=0.3,random_state=42)
```

```
[ ]  from sklearn.preprocessing import StandardScaler
     stand= StandardScaler()
     Fit = stand.fit(xtrain)
     xtrain_scl = Fit.transform(xtrain)
     xtest_scl = Fit.transform(xtest)
```

```
     # import linear regression models
     from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_squared_error
     lr=LinearRegression()
     fit=lr.fit(xtrain_scl,ytrain)
     score = lr.score(xtest_scl,ytest)
     print('predicted score is : {}'.format(score))
     print('..................................')
     y_predict = lr.predict(xtest_scl)
     print('mean_sqrd_error is ==',mean_squared_error(ytest,y_predict))
     rms = np.sqrt(mean_squared_error(ytest,y_predict))
     print('root mean squared error is == {}'.format(rms))
```

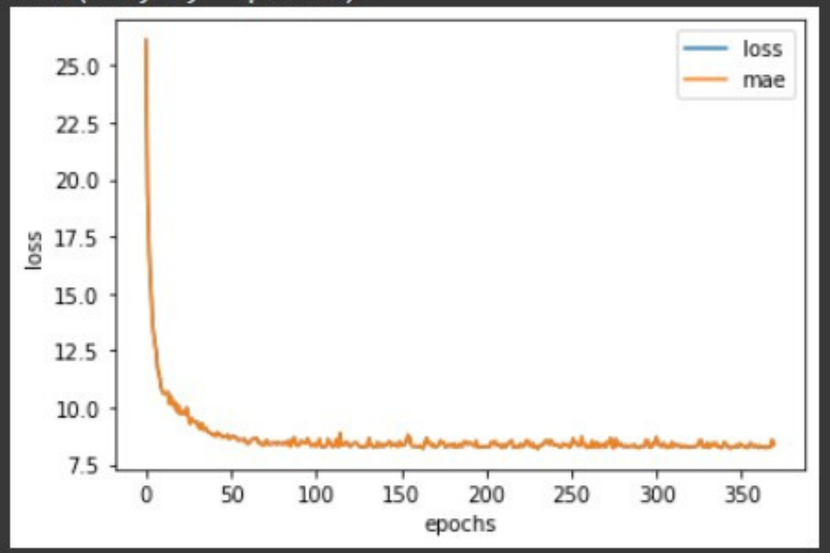
LINEAR REGRESSION

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=0.2, random_state=42)
y_test
```

```
31     52.91
109    55.90
136    74.50
88     35.30
918    10.54
       ...
482    56.14
545    18.75
110    38.00
514    74.36
602    35.17
Name: Cc_Strength, Length: 206, dtype: float64
```

```
pd.DataFrame(history.history).plot()
plt.ylabel("loss")
plt.xlabel("epochs")
```

```
Text(0.5, 0, 'epochs')
```




```
tf.random.set_seed(42)

# STEP1: Creating the model

model= tf.keras.Sequential([
                            tf.keras.layers.Dense(8),
                            tf.keras.layers.Dense(4),
                            tf.keras.layers.Dense(2),
                            tf.keras.layers.Dense(1)
])

# STEP2: Compiling the model   # optimizer casn be SGD, Adam

model.compile(loss= tf.keras.losses.mae,
              optimizer= tf.keras.optimizers.Adam(),
              metrics= ["mae"])

# STEP3: Fit the model

history= model.fit(x_train, y_train, epochs= 370, verbose=0)
```

```
[ ] model.evaluate(x_test, y_test)
```

```
7/7 [==============================] - 0s 2ms/step - loss: 7.9454 - mae: 7.9454
[7.945353031158447, 7.945353031158447]
```

NEURAL NETWORKS

```
[ ] from sklearn.ensemble import BaggingRegressor
```

```
bag_model = BaggingRegressor(
base_estimator=BaggingRegressor(),
n_estimators=500,
max_samples=0.8,
bootstrap=True,
oob_score=True,
random_state=42
)
```

```
[ ] l=bag_model.fit(x_train, y_train)
```

```
[ ] mae = metrics.mean_absolute_error(y_test, l.predict(x_test))
    print("The mean abs error (MAE) on test set: {:.10f}".format(mae))

    The mean abs error (MAE) on test set: 4.3228750661
```

BAGGING

```
from sklearn import datasets, ensemble
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error
```

```
[ ] regressor = GradientBoostingRegressor(
        max_depth=5,
        n_estimators=750,
        learning_rate=0.15,
        random_state=23 )
    regressor.fit(x_train, y_train)
    y_pred = regressor.predict(x_test)
    mean_absolute_error(y_test, y_pred)
```

```
2.9265978733238587
```

GRADIENT BOOSTING

```python
from sklearn.ensemble import AdaBoostRegressor

adaclf = AdaBoostRegressor(
                           n_estimators=750,
                           learning_rate=0.15,
                           random_state=42)


adaclf.fit(x_train, y_train)
y_pred_1 = adaclf.predict(x_test)
ab=mean_absolute_error(y_test, y_pred_1)
print("mean absolute eroor",ab)
```

```
mean absolute eroor 6.47043780975161
```

ADA BOOST

```
[ ]  from xgboost import XGBRegressor
     clf = XGBRegressor(n_estimators=750,
                        learning_rate=0.15,
                        max_depth=5,
                        random_state=23)
     clf.fit(x_train, y_train)
     y_pred = clf.predict(x_test)
     eg=mean_absolute_error(y_test, y_pred)
     print("mean absolute error",eg)
```

```
[10:46:16] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
mean absolute error 2.8415557635872104
```

EXTREME GRADIENT BOOSTING