CONCRETE STRENGTH PREDICTION

VISHAL
ZEESHAN
ISHIKA
AJINKYA
HARDEEP

DATASET SIZE

```
[ ]  #Check dataset size
     df.shape

     (1030, 9)
```

Check column types and describe which columns are numerical, or categorical

```
#Check column types and describe which columns are numerical, or categorical
df.dtypes
```

```
cement              float64
slag                float64
flyash              float64
water               float64
superplasticizer    float64
coarseaggregate     float64
fineaggregate       float64
age                   int64
csMPa               float64
dtype: object
```

Perform Univariate analysis
Calculate mean, median, std. dev and quartiles of numerical data

```
[ ] df.describe()
```

	cement	slag	flyash	water	superplasticizer	coarseaggregate	fineaggregate	age	csMPa
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	135.948932	120.899417	181.567282	9.663495	972.918932	773.580485	45.662136	35.817961
std	104.506364	53.279837	22.595744	21.354219	3.645923	77.753954	80.175980	63.169912	16.705742
min	102.000000	11.000000	24.500000	121.800000	1.700000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	129.800000	121.400000	164.900000	8.200000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	135.700000	121.400000	185.000000	9.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.950000	121.400000	192.000000	10.200000	1029.400000	824.000000	56.000000	46.135000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	365.000000	82.600000

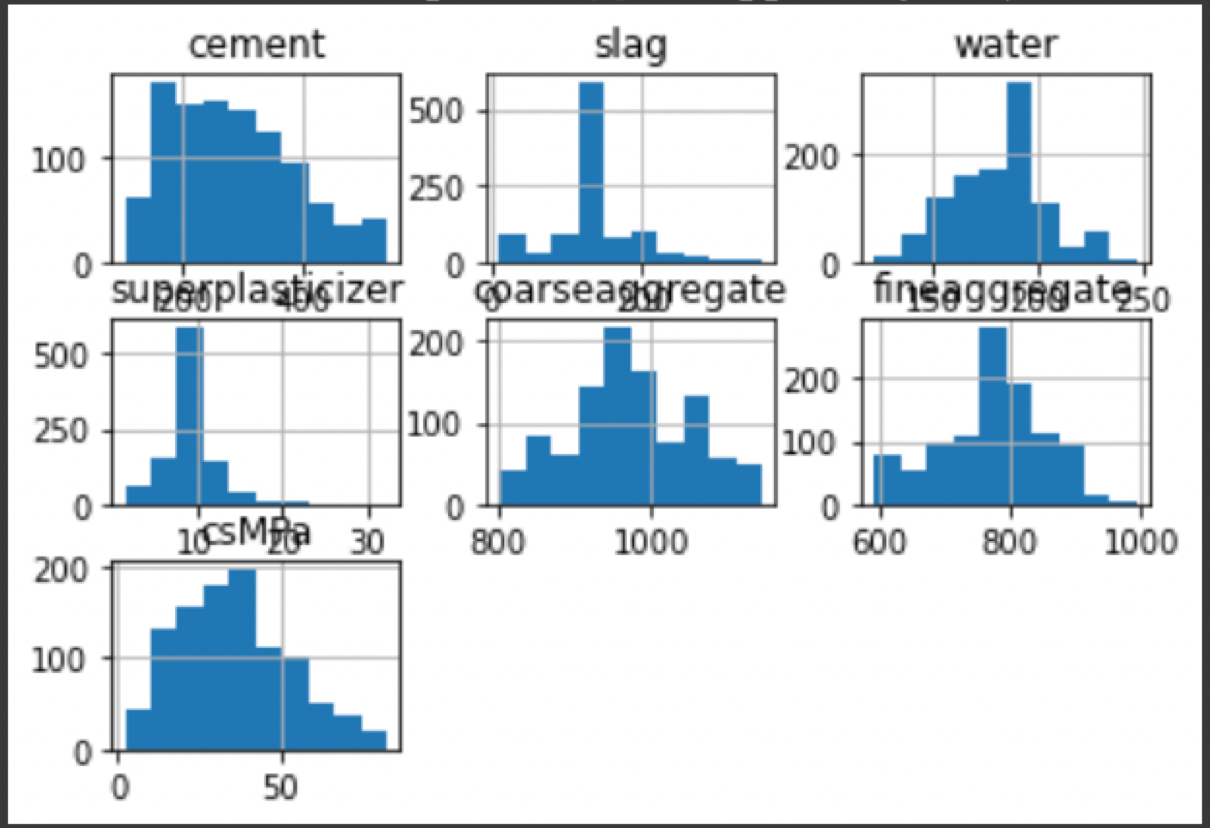
```
df.median()
```

```
cement              272.900
slag                135.700
flyash              121.400
water               185.000
superplasticizer      9.400
coarseaggregate     968.000
fineaggregate       779.500
age                  28.000
csMPa                34.445
dtype: float64
```

DISTRIBUTION OF NUMERICAL VARIABLES

```
#Check the distribution of numerical variables and comment on it
columns = ['cement', 'slag', 'water', 'superplasticizer','coarseaggregate','fineaggregate','csMPa']
df[columns].hist()
```

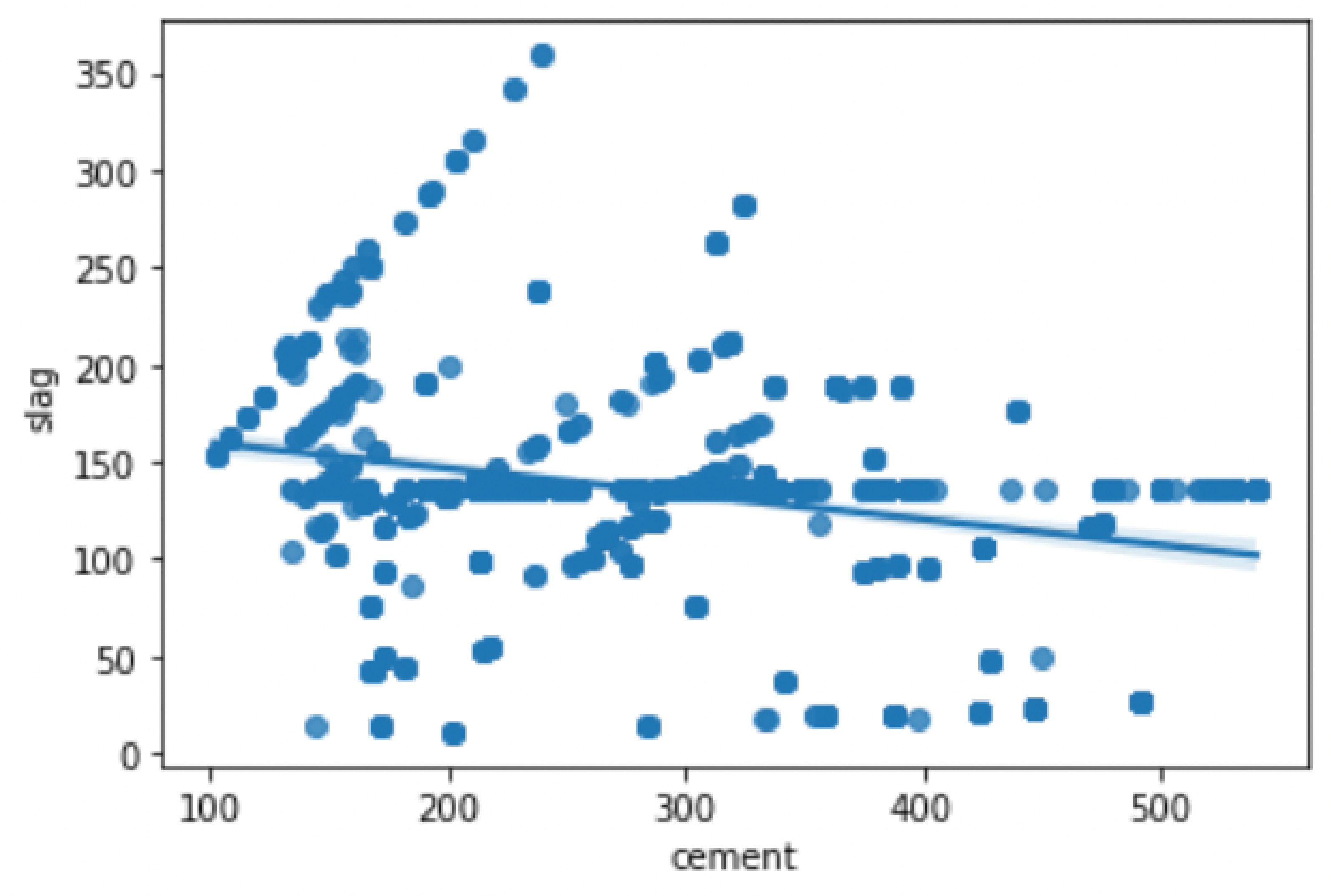
```
array([[<AxesSubplot:title={'center':'cement'}>,
        <AxesSubplot:title={'center':'slag'}>,
        <AxesSubplot:title={'center':'water'}>],
       [<AxesSubplot:title={'center':'superplasticizer'}>,
        <AxesSubplot:title={'center':'coarseaggregate'}>,
        <AxesSubplot:title={'center':'fineaggregate'}>],
       [<AxesSubplot:title={'center':'csMPa'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



PERFORM BIVARIATE ANALYSIS

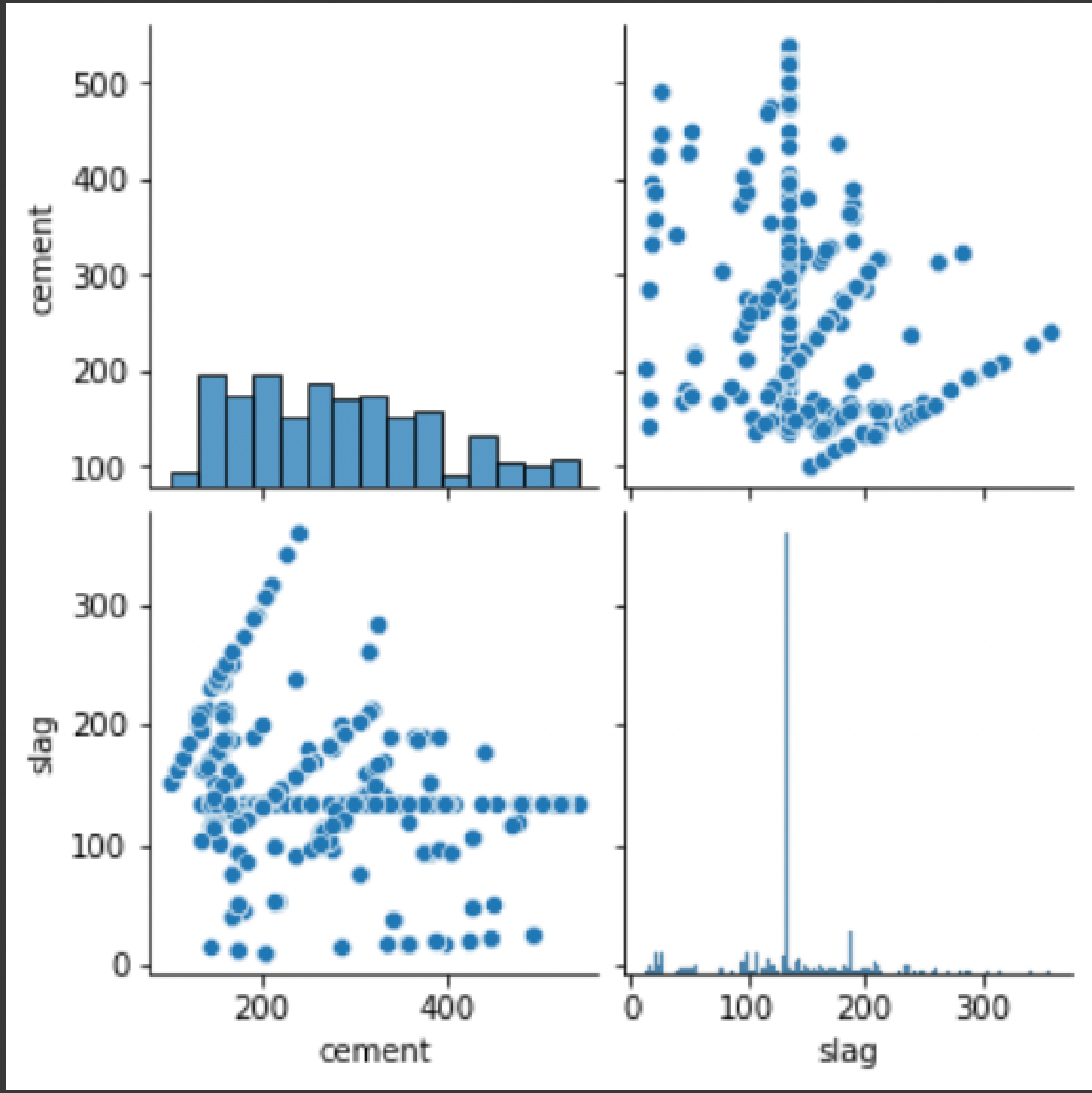
```
sb.regplot(x="cement", y="slag", data=df)
```

<AxesSubplot:xlabel='cement', ylabel='slag'>




```
[ ] #Perform Bivariate analysis
    #Plot pair plots
    sb.pairplot(df[['cement','slag']])
```

<seaborn.axisgrid.PairGrid at 0x217b5bcd3a0>



Perform Chi - square to check whether there is a relationship between age and csMPa

```python
#Perform Chi-square analysis to check whether there is a relationship between
#age and csMPa
# create contingency table
data_crosstab = pd.crosstab(df['age'],
                            df['csMPa'],
                           margins=True, margins_name="Total")

# significance level
alpha = 0.05

# Calcualtion of Chisquare
chi_square = 0
rows = df['age'].unique()
columns = df['csMPa'].unique()
for i in columns:
    for j in rows:
        O = data_crosstab[i][j]
        E = data_crosstab[i]['Total'] * data_crosstab['Total'][j] / data_crosstab['Total']['Total']
        chi_square += (O-E)**2/E

# The p-value approach
print("Approach 1: The p-value approach to hypothesis testing in the decision rule")
p_value = 1 - stats.chi2.cdf(chi_square, (len(rows)-1)*(len(columns)-1))
conclusion = "Failed to reject the null hypothesis."
if p_value <= alpha:
    conclusion = "Null Hypothesis is rejected."

print("chisquare-score is:", chi_square, " and p value is:", p_value)
print(conclusion)

# The critical value approach
print("\n--------------------------------------------------------------------------------------")
print("Approach 2: The critical value approach to hypothesis testing in the decision rule")
critical_value = stats.chi2.ppf(1-alpha, (len(rows)-1)*(len(columns)-1))
conclusion = "Failed to reject the null hypothesis."
if chi_square > critical_value:
    conclusion = "Null Hypothesis is rejected."

print("chisquare-score is:", chi_square, " and critical value is:", critical_value)
```

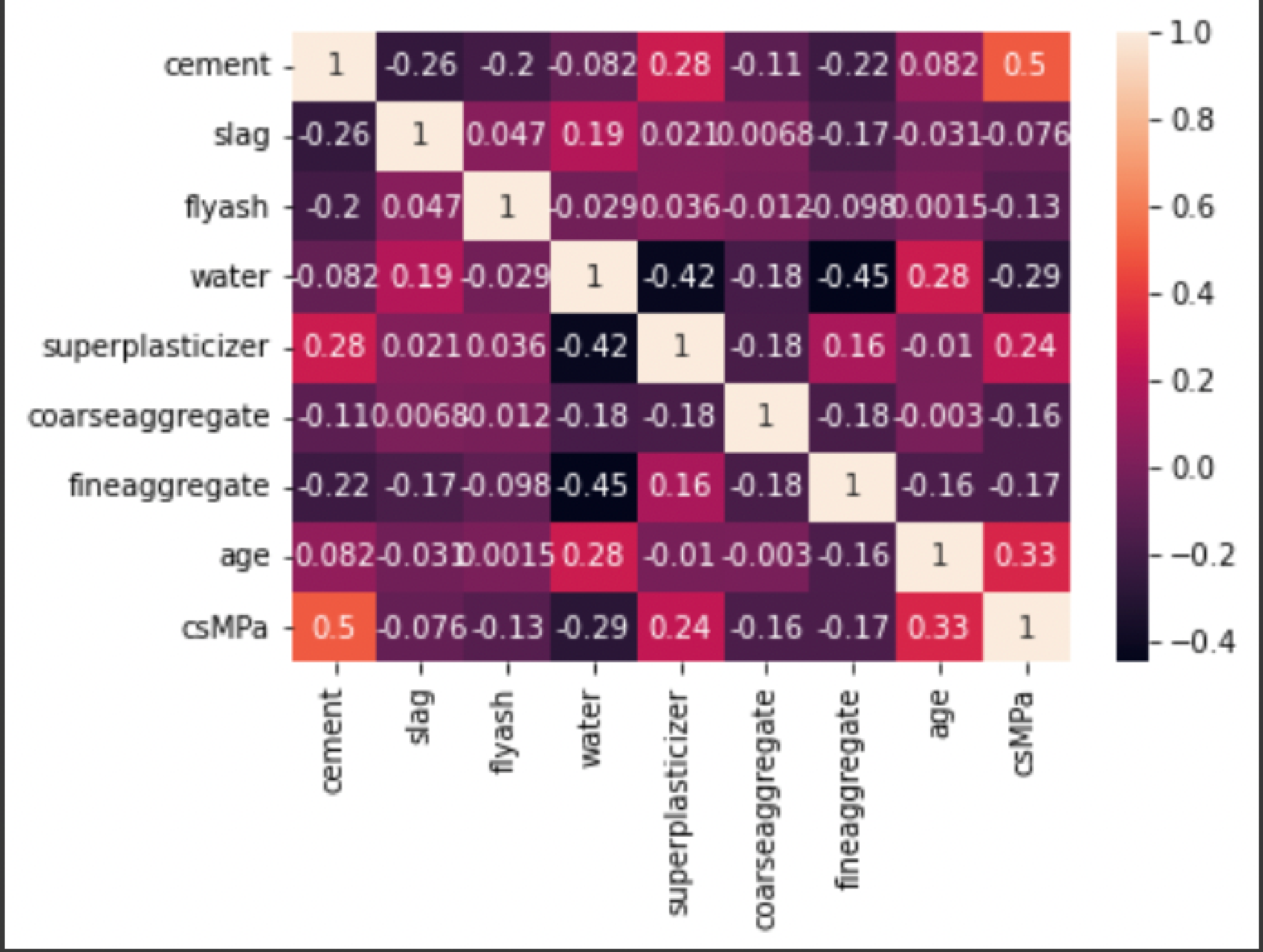
```
Approach 1: The p-value approach to hypothesis testing in the decision rule
chisquare-score is: 11462.206895002948  and p value is: 0.0005460933550068825
Null Hypothesis is rejected.

--------------------------------------------------------------------------------
Approach 2: The critical value approach to hypothesis testing in the decision rule
chisquare-score is: 11462.206895002948  and critical value is: 11216.79223246852
Null Hypothesis is rejected.
```

PEARSON CORRELATION AND HEATMAP

```
#Calculate Pearson correlation, and plot their heatmap
sb.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```



```
X=df.iloc[:,:-1].values
Y=df.iloc[:,-1].values
```

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=4)
```

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
```

```
[ ] reg.coef_
```

```
array([12.30190986,  8.78353068,  4.64335288, -3.12030687,  2.8808571 ,
        1.43649627,  1.49727268,  7.05030577])
```

```
[ ] reg.intercept_
```

```
35.61218673218672
```

```
[ ] Y_pred=reg.predict(X_test)
```

```
from sklearn import metrics
metrics.mean_squared_error(Y_test,Y_pred)
```

```
104.74182502079691
```

```
[ ] import numpy as np
np.sqrt(metrics.mean_squared_error(Y_test,Y_pred))
```

```
10.234345363568542
```

```
[ ] metrics.r2_score(Y_test,Y_pred)
```

```
0.6194861626865971
```

DECISION TREE

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,test_size=0.3, random_state=42)
```

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

```
dtc.fit(X_train, Y_train)
```

```
DecisionTreeClassifier()
```

```
Y_pred=dtc.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,Y_pred)
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, Y_pred)
```

```
0.0392156862745098
```

svm

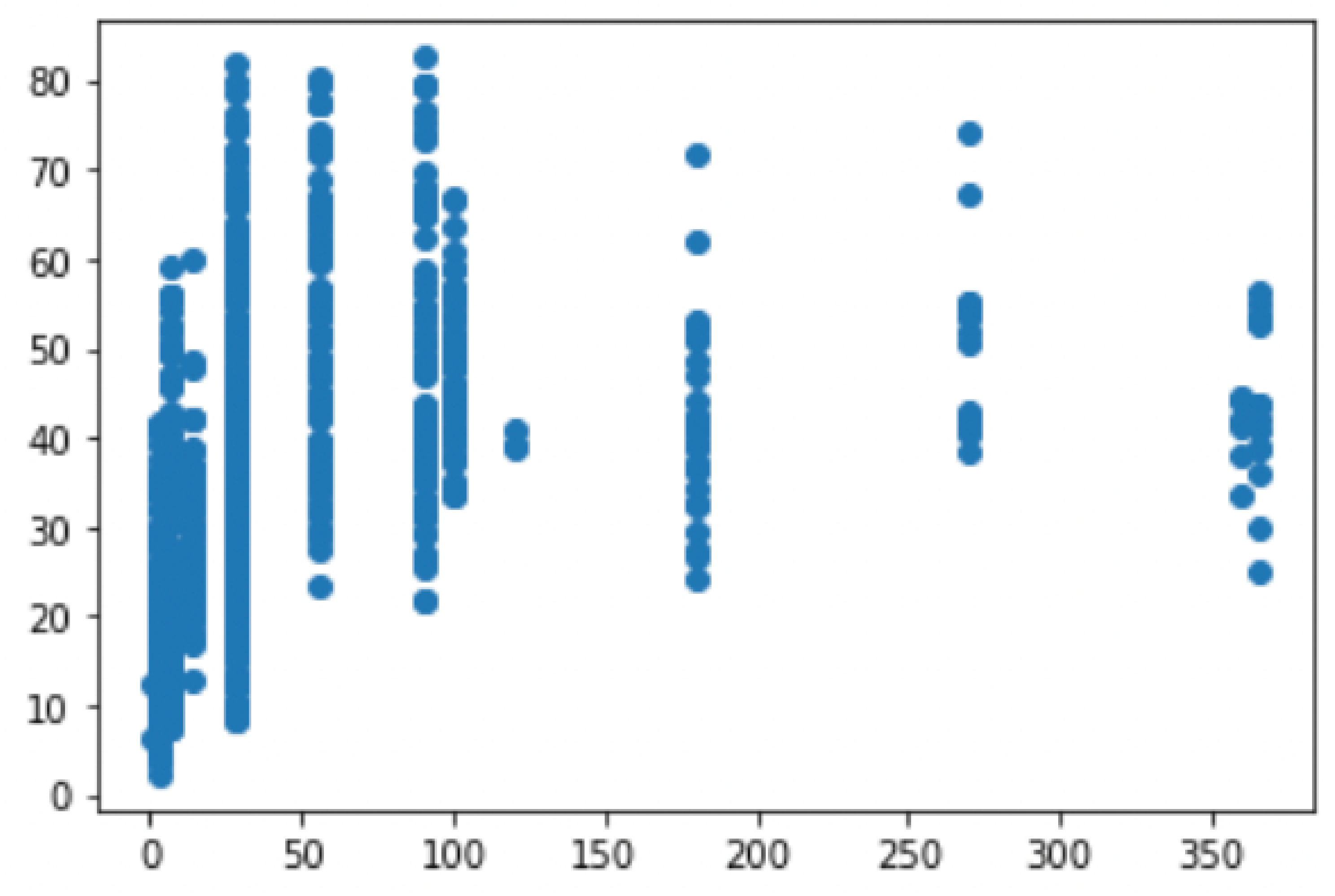
```
from sklearn.svm import SVR
```

```
regressor=SVR(kernel='linear',degree=1)
```

```
import matplotlib.pyplot as plt
```

```
plt.scatter(df['age'],df['csMPa'])
```

```
<matplotlib.collections.PathCollection at 0x217b8de4430>
```



```
df.head()
```

	cement	slag	flyash	water	superplasticizer	coarseaggregate	fineaggregate	age	csMPa
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

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,test_size=0.3, random_state=42)
```

+ Code + Text

```
regressor.fit(X_train,Y_train)
```

```
SVR(degree=1, kernel='linear')
```

```
pred=regressor.predict(X_test)
```

```
print(regressor.score(X_test,Y_test))
```

```
0.5250593257385956
```

```
[ ] from sklearn.metrics import r2_score
```

```
[ ] print(r2_score(Y_test,pred))
```

```
0.5250593257385956
```

```
regressor=SVR(kernel='rbf',epsilon=1.0)
regressor.fit(X_train,Y_train)
pred=regressor.predict(X_test)
print(regressor.score(X_test,Y_test))
print(r2_score(Y_test,pred))
```

```
0.030409950842596 79
0.030409950842596 79
```

FINDINGS

- Accuracy of linear regression - 0.619
- degree 2: 0.746
- degree 3: 0.813
- Accuracy of decision tree - 0.049
- svm - 0.525

CHALLENGES

- Data consisted of many zero values
- there were no categorical values

THANK YOU