In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# algorithms
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from scipy import stats
import math
```

Primary Exploratory Data Analysis

In [ ]:
```python
co = pd.read_csv(r'F:\concretedata.csv')
```

In [3]:
```python
co.head()
```

Out[3]:

	Cement	Blast_Furnace_Slag	Fly_Ash	Water	Superplasticizer	Coarse_Aggregate	Fine_Aggregate	Age_in_day
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	2
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	2
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	36
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	36

Changing the name of columns

In [4]:
```python
co=co.rename(columns = {"Cement":"cement","Blast_Furnace_Slag":"bf_slag","Fly_Ash":"f_
```

In [6]:
```python
co.shape
```

Out[6]:
```
(1030, 9)
```

Binder = cement + bf_slag + f_ash

In [7]:
```python
co['binder'] = co['cement'] + co['bf_slag'] + co['f_ash']
```

In [9]:
```python
co['water_binder_ratio']  = co['water'] / co['binder']
```

In [11]:
```python
co['coarse_fine_ratio'] = co['coarse_agg'] / co['fine_agg']
```

In [13]:
```python
co = co.drop(['cement','bf_slag','f_ash','water','coarse_agg','fine_agg','binder'], a
```

converting age_days into log format

In [15]:
```python
co['log_age'] = np.log(co['age_days'])
```

superplaticizer is not a binding material but act as a water ratio balancer

In [17]:
```python
sns.lineplot(x= 'water_binder_ratio', y = 'cc_strength', data =co)
```

Out[17]:
```
<AxesSubplot:xlabel='water_binder_ratio', ylabel='cc_strength'>
```

In [19]:
```python
co = co.drop(['age_days'], axis = 1)
```

binning the data into different groups

In [21]:
```python
co.describe()
```

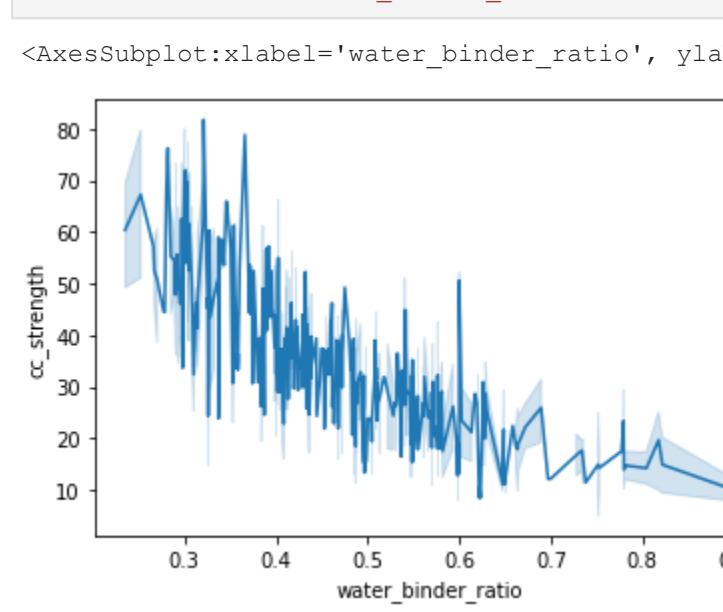
Out[21]:

	s_plasticizer	cc_strength	water_binder_ratio	coarse_fine_ratio	log_age
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	6.204660	35.817961	0.469233	1.273752	3.165326
std	5.973841	16.705742	0.127123	0.185670	1.191450
min	0.000000	2.330000	0.235073	0.858453	0.000000
25%	0.000000	23.710000	0.383916	1.121488	1.945910
50%	6.400000	34.445000	0.471974	1.266055	3.332205
75%	10.200000	46.135000	0.561081	1.358146	4.025352
max	32.200000	82.600000	0.900000	1.874876	5.899897

In [126...
```python
co2 = co.copy()
```

In [128...
```python
sns.displot(x= 's_plasticizer', data = co2, kind = 'hist')
```

Out[128...
```
<seaborn.axisgrid.FacetGrid at 0x287c0d136a0>
```



In [27]:
```python
statistic,bin_edges,binnumber = stats.binned_statistic(co2['s_plasticizer'], co2['s_p
co2['sp_bin'] = pd.DataFrame(zip(bin_edges), columns=['sp_bin'])
```

In [30]:
```python
statistic,bin_edges,binnumber = stats.binned_statistic(co2['cc_strength'], co2['cc_st
co2['cc_bin'] = pd.DataFrame(zip(bin_edges), columns=['cc_bin'])
```

In [31]:
```python
statistic,bin_edges,binnumber = stats.binned_statistic(co2['water_binder_ratio'], co2
co2['wb_bin'] = pd.DataFrame(zip(bin_edges), columns=['wb_bin'])
```

In [32]:
```python
statistic,bin_edges,binnumber = stats.binned_statistic(co2['coarse_fine_ratio'], co2[
co2['cf_bin'] = pd.DataFrame(zip(bin_edges), columns=['cf_bin'])
```

In [33]:
```python
statistic,bin_edges,binnumber = stats.binned_statistic(co2['log_age'], co2['log_age'],
co2['lg_bin'] = pd.DataFrame(zip(bin_edges), columns=['lg_bin'])
```

In [36]:
```python
co2 = co2.drop(['s_plasticizer','cc_strength','water_binder_ratio','coarse_fine_ratio
```

MODEL NO 1

In [39]:
```python
train , test = train_test_split(co2, test_size = 0.3 , random_state = 78 , shuffle = T
```

In [42]:
```python
x_train = train.drop(['cc_bin'], axis = 1)
y_train = train['cc_bin']
x_test = test.drop(['cc_bin'], axis = 1)
y_test = test['cc_bin']
```

In [45]:
```python
num_col = [ x for x in x_train.columns]

for i in num_col:
    scale = StandardScaler().fit(x_train[[i]])

    x_train[i] = scale.transform(x_train[[i]])

    x_test[i]  = scale.transform(x_test[[i]])
```

In [50]:
```python
LR = LinearRegression()
model_12 = LR.fit(x_train,y_train)
```

PERFORMANCE of model no 1

In [51]:
```python
# train
y_pred_train = model_12.predict(x_train)

print("MSE" , metrics.mean_squared_error(y_train,y_pred_train))
print("R squared",metrics.r2_score(y_train,y_pred_train))
```

```
MSE 4.4487214454247777e-29
R squared 1.0
```

In [52]:
```python
# test
y_pred_test = model_12.predict(x_test)

print("MSE" , metrics.mean_squared_error(y_test,y_pred_test))
print("R squared",metrics.r2_score(y_test,y_pred_test))
```

```
MSE 4.891831144139941e-29
R squared 1.0
```

In [53]:
```python
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
print('MAPE:', mean_absolute_percentage_error(y_test, y_pred_test))
```

```
MAPE: 3.064408755900244e-14
```

Data Preprocessing for Model NO 2

In [80]:
```python
co3 = co.copy()
```

binning the data and taking mean as statistics

In [83]:
```python
x = co3['s_plasticizer']
```

In [86]:
```python
x = np.sort(co3['s_plasticizer'])
```

In [87]:
```python
print(x)
```

```
[ 0.   0.   0.  ... 32.2 32.2 32.2]
```

converting into bins

In [88]:
```python
bin1 = np.zeros((10,103))
```

In [89]:
```python
print(bin1)
```

```
[[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.]]
```

In [90]:
```python
print(type(bin1))
```

```
<class 'numpy.ndarray'>
```

In [91]:
```python
print(type(x))
```

```
<class 'numpy.ndarray'>
```

In [93]:
```python
print(x)
```

```
[ 0.   0.   0.  ... 32.2 32.2 32.2]
```

x numpy array

In [94]:
```python
print(x)
```

```
[ 0.   0.   0.  ... 32.2 32.2 32.2]
```

convert into 20 bins

In [113...
```python
bin1 = np.zeros((206,5))
```

In [114...
```python
print(bin1)
```

```
[[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.]]
```

In [115...
```python
for i in range(0,1030,5):
    k = int(i/5)
    mean = (x[i] + x[i+1] + x[i+2] + x[i+3] + x[i+4])/5
    for j in range(5):
        bin1[k,j] = mean
print("Bin Mean: \n",bin1)
```

```
Bin Mean:
 [[ 0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0. ]
 ...
 [23.4 23.4 23.4 23.4 23.4]
 [28.2 28.2 28.2 28.2 28.2]
 [32.2 32.2 32.2 32.2 32.2]]
```

In [116...
```python
print(type(bin1))
```

```
<class 'numpy.ndarray'>
```

convert numpy array into pandas series

In [119...
```python
one_array =  bin1.flatten()
```

In [120...
```python
print(one_array)
```

```
[ 0.   0.   0.  ... 32.2 32.2 32.2]
```

In [201...
```python
s_plasticizer = pd.Series(one_array)
```

In [202...
```python
print(s_plasticizer)
```

```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
        ...
1025    32.2
1026    32.2
1027    32.2
1028    32.2
1029    32.2
Length: 1030, dtype: float64
```

In [131...
```python
x1 = co2['cc_strength']
```

In [133...
```python
print(type(x1))
```

```
<class 'pandas.core.series.Series'>
```

In [134...
```python
x1 = x1.to_numpy()
```

In [135...
```python
print(type(x1))
```

```
<class 'numpy.ndarray'>
```

In [136...
```python
x1 = np.sort(x1)
```

In [137...
```python
print(x1)
```

```
[ 2.33  3.32  4.57 ... 80.2  81.75 82.6 ]
```

In [138...
```python
bin2 = np.zeros((206,5))
```

In [139...
```python
print(bin2)
```

```
[[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.]]
```

In [141...
```python
for i in range(0, 1030 , 5):
    k = int(i/5)
    mean = (x1[i] + x1[i+1] + x1[i+2] + x1[i+3] + x1[i+4])/5
    for j in range(5):
        bin2[k,j] = mean
print("bin mean \n", bin2)
```

```
bin mean
 [[ 3.966  3.966  3.966  3.966  3.966]
 [ 6.146  6.146  6.146  6.146  6.146]
 [ 7.088  7.088  7.088  7.088  7.088]
 ...
 [77.2   77.2   77.2   77.2   77.2  ]
 [79.2   79.2   79.2   79.2   79.2  ]
 [80.788 80.788 80.788 80.788 80.788]]
```

In [142...
```python
bin2 = bin2.flatten()
```

In [143...
```python
print(bin2)
```

```
[ 3.966  3.966  3.966 ... 80.788 80.788 80.788]
```

In [199...
```python
cc_strength = pd.Series(bin2)
```

In [200...
```python
print(cc_strength)
```

```
0        3.966
1        3.966
2        3.966
3        3.966
4        3.966
         ...
1025    80.788
1026    80.788
1027    80.788
1028    80.788
1029    80.788
Length: 1030, dtype: float64
```

water_binder_ratio

In [146...
```python
x2 = co2['water_binder_ratio']
```

In [147...
```python
x2 = x2.to_numpy()
```

In [150...
```python
x2 = np.sort(x2)
```

In [151...
```python
print(type(x2))
```

```
<class 'numpy.ndarray'>
```

In [152...
```python
bin3 = np.zeros((206,5))
```

In [153...
```python
print(bin3)
```

```
[[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.]]
```

In [156...
```python
for i in range(0,1030,5):
    k = int(i/5)
    mean = (x2[i] + x2[i+1] + x2[i+2] + x2[i+3] + x2[i+4])/5
    for j in range(5):
        bin3[k,j] = mean
print("bin3 mean \n" , bin3)
```

```
bin3 mean
 [[0.23507335 0.23507335 0.23507335 0.23507335 0.23507335]
 [0.25202971 0.25202971 0.25202971 0.25202971 0.25202971]
 [0.26669754 0.26669754 0.26669754 0.26669754 0.26669754]
 ...
 [0.78546369 0.78546369 0.78546369 0.78546369 0.78546369]
 [0.81597363 0.81597363 0.81597363 0.81597363 0.81597363]
 [0.85322034 0.85322034 0.85322034 0.85322034 0.85322034]]
```

In [157...
```python
bin3 = bin3.flatten()
```

In [158...
```python
print(bin3)
```

```
[0.23507335 0.23507335 0.23507335 ... 0.85322034 0.85322034 0.85322034]
```

In [203...
```python
water_binder_ratio = pd.Series(bin3)
```

In [204...
```python
print(water_binder_ratio)
```

```
0       0.235073
1       0.235073
2       0.235073
3       0.235073
4       0.235073
          ...
1025    0.853220
1026    0.853220
1027    0.853220
1028    0.853220
1029    0.853220
Length: 1030, dtype: float64
```

coarse fine ratio :

In [164...
```python
x4 = co2['coarse_fine_ratio']
```

In [165...
```python
x4 = x4.to_numpy()
```

In [167...
```python
print(type(x4))
```

```
<class 'numpy.ndarray'>
```

In [168...
```python
x4 = np.sort(x4)
```

In [169...
```python
bin4 = np.zeros((206,5))
```

In [170...
```python
print(bin4)
```

```
[[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.]]
```

In [171...
```python
for i in range(0,1030,5):
    k = int(i/5)
    mean = (x4[i]+ x4[i+1] + x4[i+2] + x4[i+3] + x4[i+4])/5
    for j in range(5):
        bin4[k,j] = mean
```

In [172...
```python
print(bin4)
```

```
[[0.85845255 0.85845255 0.85845255 0.85845255 0.85845255]
 [0.9204926  0.9204926  0.9204926  0.9204926  0.9204926 ]
 [0.9407463  0.9407463  0.9407463  0.9407463  0.9407463 ]
 ...
 [1.83523654 1.83523654 1.83523654 1.83523654 1.83523654]
 [1.83523654 1.83523654 1.83523654 1.83523654 1.83523654]
 [1.87487603 1.87487603 1.87487603 1.87487603 1.87487603]]
```

In [173...
```python
bin4 = bin4.flatten()
```

In [205...
```python
coarse_fine_ratio = pd.Series(bin4)
```

In [206...
```python
print(coarse_fine_ratio)
```

```
0       0.858453
1       0.858453
2       0.858453
3       0.858453
4       0.858453
          ...
1025    1.874876
1026    1.874876
1027    1.874876
1028    1.874876
1029    1.874876
Length: 1030, dtype: float64
```

log_age

In [176...
```python
x5 = co2['log_age']
```

In [177...
```python
x5 =x5.to_numpy()
```

In [178...
```python
print(type(x5))
```

```
<class 'numpy.ndarray'>
```

In [179...
```python
x5 = np.sort(x5)
```

In [184...
```python
print(x5)
```

```
[0.         0.         1.09861229 ... 5.89989735 5.89989735 5.89989735]
```

In [180...
```python
bin5 = np.zeros((206,5))
```

In [182...
```python
for i in range(0,1030,5):
    k  = int(i/5)
    mean = (x5[i]  + x5[i+1] + x5[i+2] + x5[i+3] + x5[i+4])/5
    for j in range(5):
        bin5[k,j] = mean
```

In [183...
```python
print(bin5)
```

```
[[0.65916737 0.65916737 0.65916737 0.65916737 0.65916737]
 [1.09861229 1.09861229 1.09861229 1.09861229 1.09861229]
 [1.09861229 1.09861229 1.09861229 1.09861229 1.09861229]
 ...
 [5.89713869 5.89713869 5.89713869 5.89713869 5.89713869]
 [5.89989735 5.89989735 5.89989735 5.89989735 5.89989735]
 [5.89989735 5.89989735 5.89989735 5.89989735 5.89989735]]
```

In [185...
```python
bin5 = bin5.flatten()
```

In [186...
```python
print(bin5)
```

```
[0.65916737 0.65916737 0.65916737 ... 5.89989735 5.89989735 5.89989735]
```

In [207...
```python
log_age = pd.Series(bin5)
```

In [208...
```python
print(log_age)
```

```
0       0.659167
1       0.659167
2       0.659167
3       0.659167
4       0.659167
          ...
1025    5.899897
1026    5.899897
1027    5.899897
1028    5.899897
1029    5.899897
Length: 1030, dtype: float64
```

combinning all these dataframe into pandasdataframe

In [189...
```python
co3 = pd.DataFrame(series_1)
```

In [190...
```python
print(co3)
```

```
         0
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
...    ...
1025  32.2
1026  32.2
1027  32.2
1028  32.2
1029  32.2

[1030 rows x 1 columns]
```

In [209...
```python
data = {"s_plasticizer": s_plasticizer, "cc_strength": cc_strength,"water_binder_ratio
```

In [210...
```python
co4 = pd.concat(data, axis =1)
```

In [213...
```python
co4.shape
```

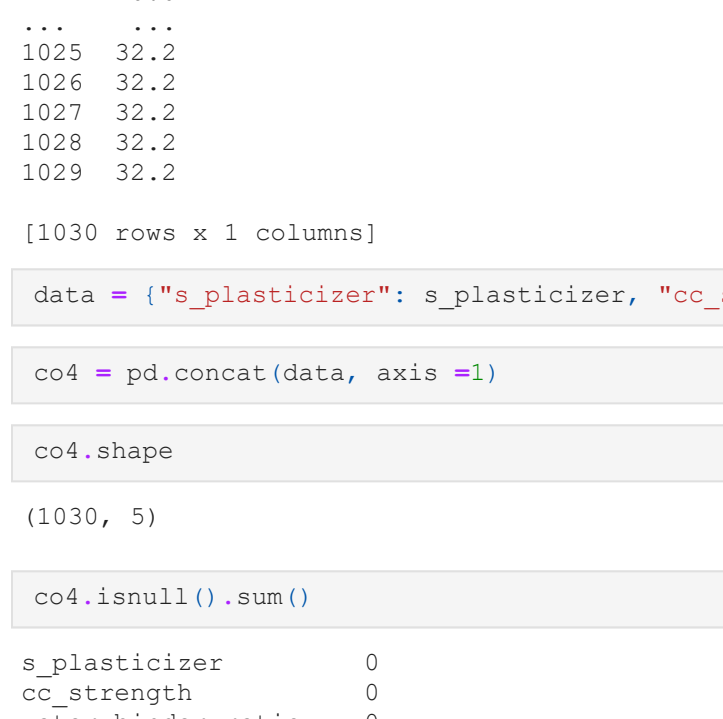
Out[213...
```
(1030, 5)
```

In [214...
```python
co4.isnull().sum()
```

Out[214...
```
s_plasticizer         0
cc_strength           0
water_binder_ratio    0
coarse_fine_ratio     0
log_age               0
dtype: int64
```

In [219...
```python
sns.displot(x= 'log_age', data = co4 , kind = 'hist')
```

Out[219...
```
<seaborn.axisgrid.FacetGrid at 0x287c11e6910>
```



MODEL NO 2

In [220...
```python
train_1 , test_1 = train_test_split(co4, test_size = 0.3 , random_state = 68 , shuffle
```

In [222...
```python
x_train = train_1.drop(['cc_strength'], axis = 1)
y_train = train_1['cc_strength']
x_test = test_1.drop(['cc_strength'], axis = 1)
y_test = test_1['cc_strength']
```

In [223...
```python
num_col = [ x for x in x_train.columns]

for i in num_col:
    scale = StandardScaler().fit(x_train[[i]])

    x_train[i] = scale.transform(x_train[[i]])

    x_test[i]  = scale.transform(x_test[[i]])
```

In [229...
```python
LR = LinearRegression()
model_34 = LR.fit(x_train,y_train)
```

Performance of model no 2

In [230...
```python
# train
y_pred_train = model_34.predict(x_train)

print("MSE" , metrics.mean_squared_error(y_train,y_pred_train))
print("R squared",metrics.r2_score(y_train,y_pred_train))
```

```
MSE 2.3366510619689884
R squared 0.9916149168413269
```

In [231...
```python
# test
y_pred_test = model_34.predict(x_test)

print("MSE" , metrics.mean_squared_error(y_test,y_pred_test))
print("R squared",metrics.r2_score(y_test,y_pred_test))
```

```
MSE 2.3923292904040885
R squared 0.991418680756592
```

In [232...
```python
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
print('MAPE:', mean_absolute_percentage_error(y_test, y_pred_test))
```

```
MAPE: 4.166966965309672
```

NOTE : Both model got more than 99% accuracy in test data

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: