Shrey Kshatriya Lab 4

November 4, 2020

1 BIA 656

1.1 Advance Data Analytics and Machine Learning

1.2 Assignment : Lab 4

1.3 Shrey Kshatriya

2 Q1. Load the CSV data into a pandas data frame. Print some high-level statistical info about the data frame's columns.

[7]:
```
import pandas as pd
import numpy as np

dx = pd.read_csv('Concrete_Data.csv')
dx.head()
```

[7]:

	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

You can use 'describe' for additional information as it gives the mean, std and IQR values.

[8]:
```
dx.describe()
```

[8]:

	Cement	Blast_Furnace_Slag	Fly_Ash	Water
count	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282
std	104.506364	86.279342	63.997004	21.354219

```
min     102.000000            0.000000     0.000000   121.800000
25%     192.375000            0.000000     0.000000   164.900000
50%     272.900000           22.000000     0.000000   185.000000
75%     350.000000          142.950000   118.300000   192.000000
max     540.000000          359.400000   200.100000   247.000000

       Superplasticizer  Coarse_Aggregate  Fine_Aggregate          Age  \
count       1030.000000       1030.000000    1030.000000  1030.000000
mean           6.204660        972.918932     773.580485    45.662136
std            5.973841         77.753954      80.175980    63.169912
min            0.000000        801.000000     594.000000     1.000000
25%            0.000000        932.000000     730.950000     7.000000
50%            6.400000        968.000000     779.500000    28.000000
75%           10.200000       1029.400000     824.000000    56.000000
max           32.200000       1145.000000     992.600000   365.000000

       Concrete_Compressive_Strength
count                    1030.000000
mean                       35.817961
std                        16.705742
min                         2.330000
25%                        23.710000
50%                        34.445000
75%                        46.135000
max                        82.600000
```

To get high level information, you can use 'info'

[9]:
```
dx.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
Cement                          1030 non-null float64
Blast_Furnace_Slag              1030 non-null float64
Fly_Ash                         1030 non-null float64
Water                           1030 non-null float64
Superplasticizer                1030 non-null float64
Coarse_Aggregate                1030 non-null float64
Fine_Aggregate                  1030 non-null float64
Age                             1030 non-null int64
Concrete_Compressive_Strength   1030 non-null float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

3 Q2. How many rows have a compressive strength > 40 MPa?

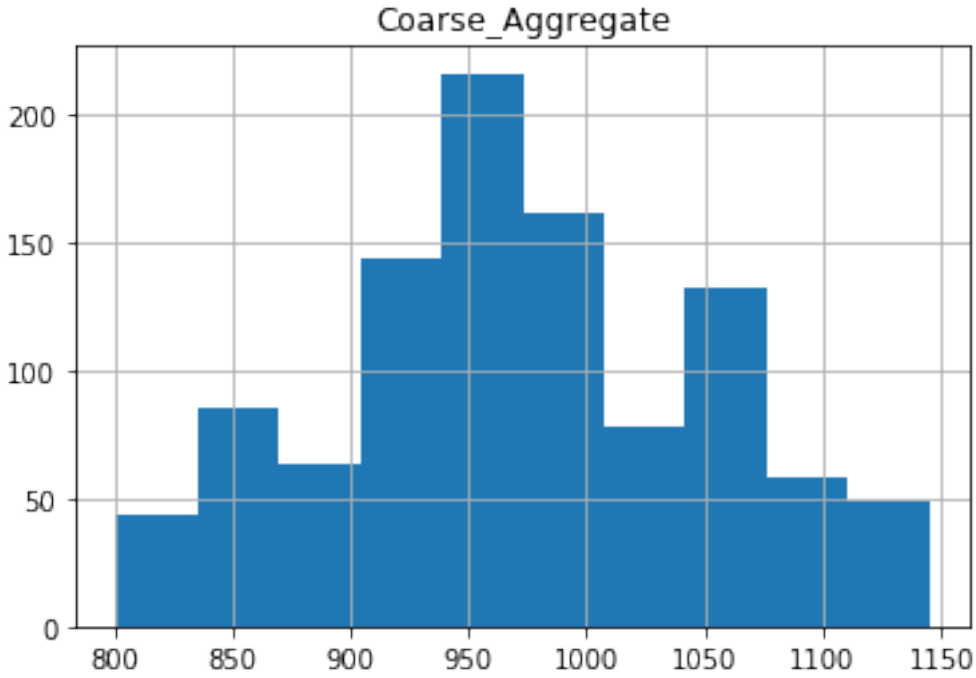
[10]:
```
len(dx[dx['Concrete_Compressive_Strength']>40])
```

[10]:
```
379
```

4 Q3. Plot the histogram of Coarse Aggregate and Fine Aggregate values

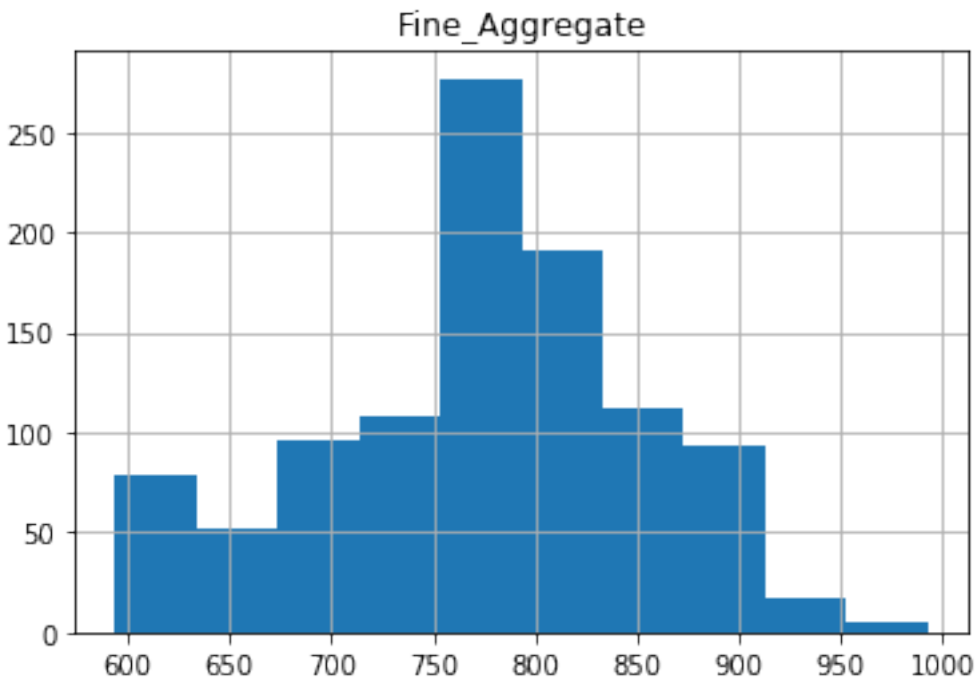
[12]:
```
dx.hist(column='Coarse_Aggregate')
```

[12]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000029DB8139278>]],
      dtype=object)
```



[13]:
```
dx.hist(column='Fine_Aggregate')
```

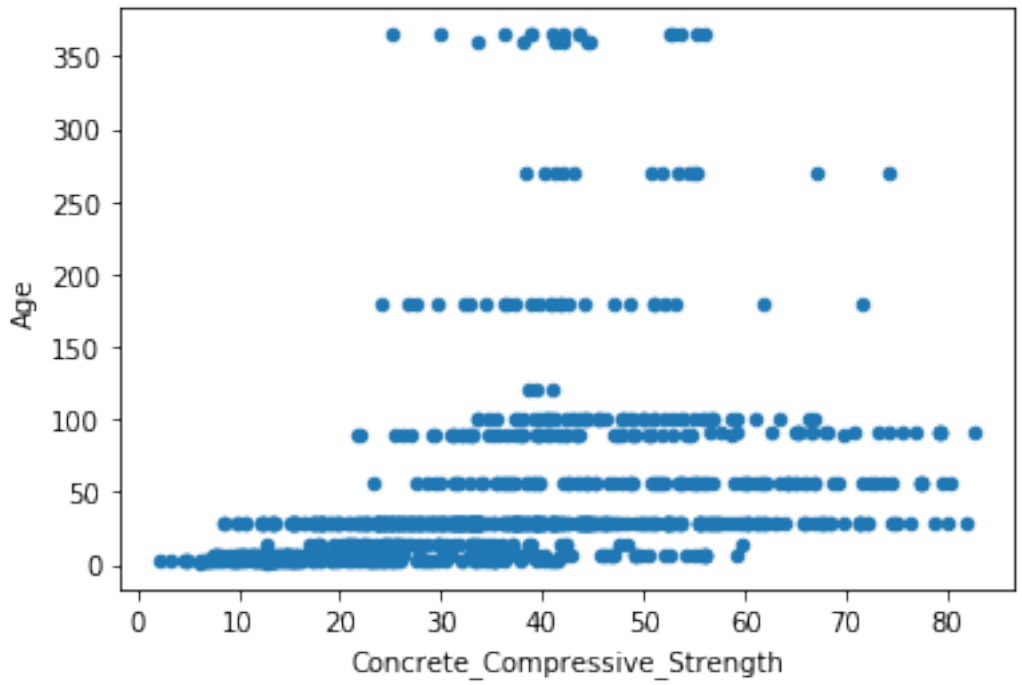
[13]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000029DB864CF98>]],
      dtype=object)
```



5 Q4. Make a plot comparing compressive strength to age

[24]:
```
dx.plot('Concrete_Compressive_Strength', 'Age', 'scatter')
```

[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29db9a29160>
```

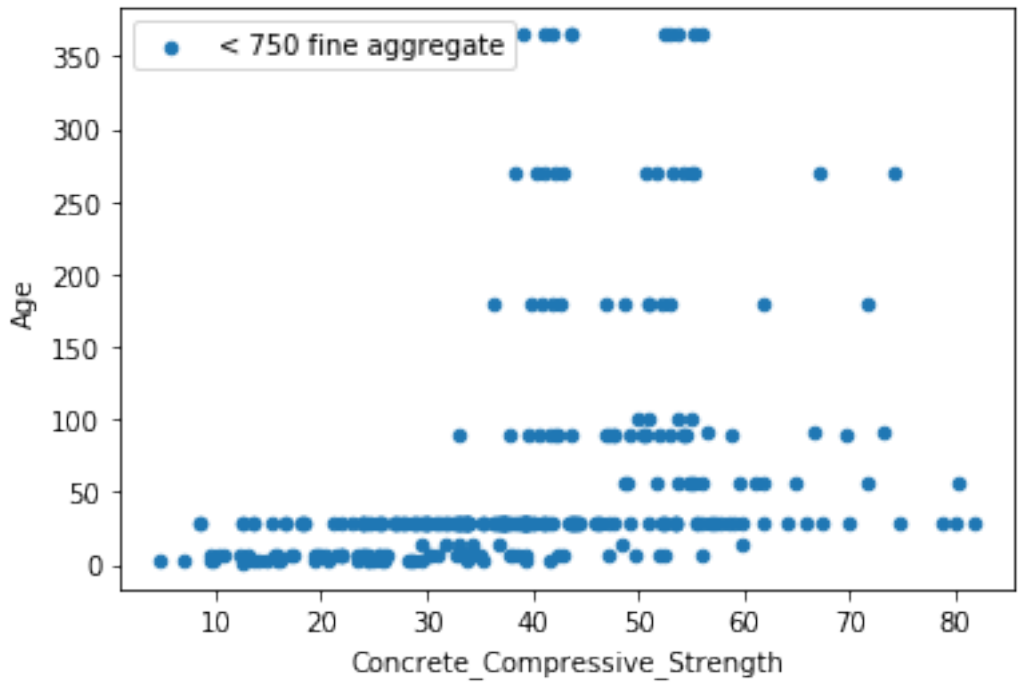


6 Q5. Make a plot comparing compressive strength to age for only those rows with < 750 fine aggregate.

To do that we create a data frame of rows with < 750 fine aggregate.

[29]:
```
temp = dx['Fine_Aggregate'] < 750
dy = dx[temp]
dy.plot('Concrete_Compressive_Strength', 'Age', 'scatter', label = '< 750 fine␣
↪aggregate')
```

[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29db9bd5438>
```



7 Q6. Try to build a linear model that predicts compressive strength given the other available fields.

[30]:
```
# First we'll need to import the predictive model we'll use
from sklearn import linear_model

# Choose a particular sort of linear regression model
# Get set up to train one of those
my_linear = linear_model.Lasso(alpha=0.01)

# Assemble the training data
# Let's use these columns as features and the target variable
features = ["Cement", "Blast_Furnace_Slag", "Fly_Ash", "Water",␣
 ↪"Superplasticizer", "Fine_Aggregate", "Age"]
target = "Concrete_Compressive_Strength"

# Eliminate (drop) any instances with missing values (NaNs) for now
cleaned_dx = dx.dropna()

# Train the model you set up on the data
#   a.k.a. Fit the model to the data!
my_linear.fit(cleaned_dx[features], cleaned_dx[target])
```

```
# Show the coefficients of the linear model
pd.DataFrame([dict(zip(features, my_linear.coef_))])
```

[30]:

	Age	Blast_Furnace_Slag	Cement	Fine_Aggregate	Fly_Ash	Superplasticizer	Water
0	0.113722	0.08836	0.107103	0.003106	0.071129	0.228159	-0.213478

8 Q7. Generate predictions for all the observations and a scatterplot comparing the predicted compressive strengths to the actual values.

[35]:
```
preds = my_linear.predict(cleaned_dx[features])
predictions_dx = cleaned_dx.assign(predictions=preds)
predictions_dx[["Concrete_Compressive_Strength", "predictions"]]
```

[35]:

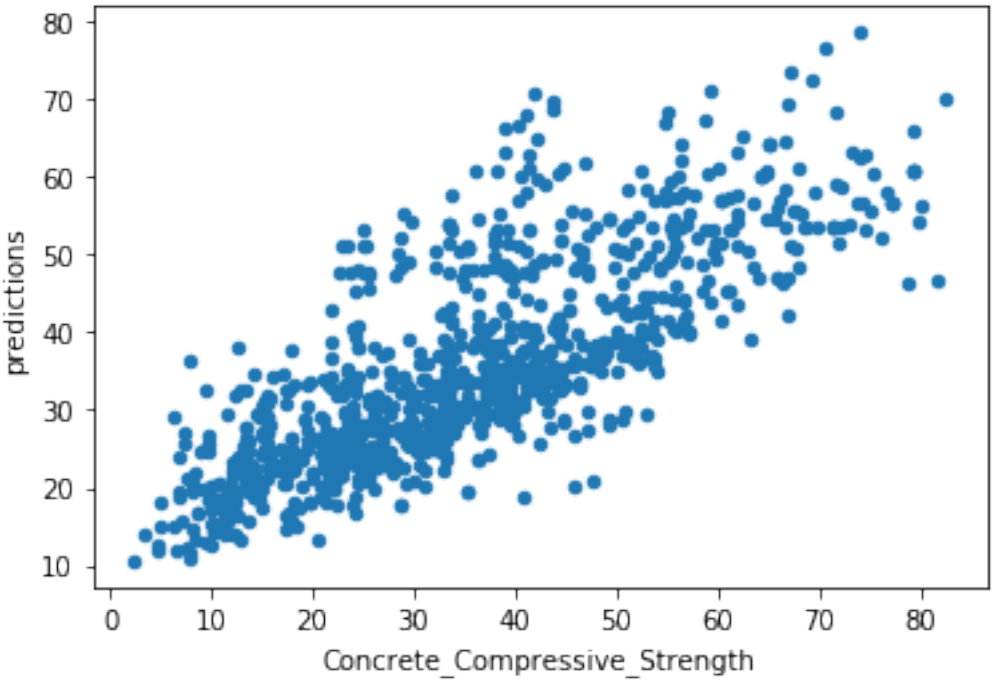
	Concrete_Compressive_Strength	predictions
0	79.99	54.176105
1	61.89	54.176105
2	40.27	57.149638
3	41.05	67.953202
4	44.30	60.555274
5	47.03	27.275164
6	43.70	68.843481
7	36.45	30.519257
8	45.85	20.224417
9	39.29	32.299813
10	38.07	29.850406
11	28.02	22.799659
12	43.01	58.930194
13	42.33	25.850719
14	47.81	20.936640
15	52.91	29.411832
16	39.36	30.277302
17	56.14	59.973086
18	40.56	37.570004
19	42.62	49.585517
20	41.84	48.695238
21	28.24	23.226555
22	8.06	20.383511
23	44.21	40.512258
24	52.52	60.685308
25	53.30	49.881744
26	41.15	58.039916
27	52.12	38.934565
28	37.43	31.409535

```
29                              38.60    29.911657
...                               ...          ...
1000                            44.61    35.285700
1001                            53.52    40.307085
1002                            57.22    39.960817
1003                            65.91    46.995345
1004                            52.83    44.616188
1005                            33.40    32.284618
1006                            18.03    22.746718
1007                            37.36    32.553130
1008                            35.31    33.840614
1009                            42.64    31.923568
1010                            40.06    30.121809
1011                            43.80    40.563066
1012                            61.24    45.168912
1013                            40.87    40.105858
1014                            33.31    37.430981
1015                            52.43    40.403776
1016                            15.09    26.893392
1017                            38.46    38.338114
1018                            37.27    37.860028
1019                            35.23    19.611015
1020                            42.14    36.136493
1021                            31.88    27.880531
1022                            41.54    34.609432
1023                            39.46    36.430425
1024                            37.92    35.336601
1025                            44.28    40.606188
1026                            31.18    34.042555
1027                            23.70    26.877963
1028                            32.77    29.332618
1029                            32.40    32.150362

[1030 rows x 2 columns]
```

[37]:
```
predictions_dx.plot('Concrete_Compressive_Strength', 'predictions', 'scatter')
```

[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29db9c41400>
```


predictions
Concrete_Compressive_Strength
80
70
60
50
40
30
20
10
0
10
20
30
40
50
60
70
80