Mitali Yadav
3034158469

Stat 154 Midterm Report

The dataset provided was about the components making up concrete and how their proportions affected the strength of the concrete. Thus, the aim of this analysis was to use 2 different ways to analyze the dataset and decide which columns are more relevant if one had to train a regression model on this dataset.

The first method used was Exploratory Data Analysis (EDA). I started with dropping the unnecessary columns since that was our ultimate goal and it would help me focus on the relevant columns. Thus I dropped columns No, Slump and Flow, which were the other output variables we were not asked to use as target variables. Next, I used a combination of the info() and describe() function to further understand the data and I made the following observations:

1. There are no null values in this dataset.
2. The data is not scaled and if I were to perform PCA on the dataset then I would have to scale and/or standardize the input variables.
3. Columns like Cement, Slag, Fly ash, Water are somewhat evenly distributed with few outliers since the 75% mark is close to the maximum value of the column. However, I decided to confirm this using boxplots.

As mentioned above, I plotted the boxplots of individual columns and as expected I got results that were not unlike the observations I made earlier. Also, most columns, except SP and the target variable are evenly distributed. While this was extremely useful to study the columns individually, they could also be plotted on a single graph since the description mentioned that all of the input variables were measured in the same unit (number of kilograms per M^3).

Next, I wanted to understand the correlation between each of the input variables as well as their relation with the target variable. I tried using 3 different plotting styles to see if I could reveal more information about the correlation plot but the lineplot was difficult to interpret. The scatter plot and kernel density plot provided me with the following insights:

1. Cement, Slag, Fly Ash all seem to have 2 peaks hence they have a bimodal distribution.
2. SP also has 2 peaks but one of them is higher than the other one.

3. Cement and Fly Ash seem to have a positive correlation with Compressive Strength (our target variable)
4. Water and Coarse Aggr. have a weak negative correlation which makes sense intuitively because the amount of water would be inversely proportional to the coarseness of the cement (a higher amount of water would result in a finer and thinner consistency).

I used the heatmap next just to get a more accurate picture of the correlations.

Based on this analysis, I would conclude that SP, coarse and fine aggregates do not have a strong correlation with the target variable although coarse aggregate did have a strong negative correlation with water thus could be used as one of the predictors.

The second method I used was based upon this hypothesis that I only needed 5 input variables to be able to accurately predict the target variable. I decided to use PCA (Principal Component Analysis.) The aim was to reduce the dimensions of the input data which would, in turn, eliminate the "noise" and make the data easier for processing and modeling for prediction.

I started by extracting the input variables, converting the dataframe into a matrix, and normalizing that matrix using StandardScaler(). This is done so as to ensure that each of the columns has equal variance and equal weightage. After that, I prepare the data by converting it back to a dataframe from the matrix. I used PCA() from the sklearn library and since I am not sure as to how many components I will end up with I initialize the number of components to 7. The sklearn's PCA is especially useful because I can easily create a scree plot by plotting the values given by the explained_variance_ratio_ method. The scree plot is yet another visual tool that used to deduce the number of components that could be selected without losing too much information. Unfortunately, there was no sharp elbow in the scree plot that could be used as a cut-off but at pc_4, we could explain 83% of the variance. This was less than the usual expected 99% but this was a trade-off I had to make to reduce the dimension of the dataset from 7 to 4. There was not as much of a dramatic reduction in dimensions because cement requires 4 key components that are water, coarse aggregates, fine aggregates, and cement. Thus we could not go lower than 4 components and I would make an educated guess that the discarded data was a combination of the rest of the materials that would alter the durability of the concrete. Components such as slag, fly ash, superplasticizer, and many others had been eliminated from this dataset, probably after conducting a similar PCA. I ended this analysis by creating a dataframe with the principal components of all the samples. Ideally, this would serve as the starting point for PC Regression and would provide more accurate results compared to the original data.

In conclusion, I have learned from this dataset that while components like SP, slag, and fly-ash play a crucial role in the durability of the concrete, there is not a lot of overlap between durability and strength, which was our target variable. I also learned that using the right data visualizations during EDA can provide valuable insights into the data and at times be just as useful as PCA when the number of input variables is low (not in the hundreds). Currently, there is not a lot of data to work with so increasing the number of rows could increase the accuracy of predictions.

MT

In [ ]:

```

```

In [15]:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
from sklearn.decomposition import PCA
from sklearn.manifold import MDS
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
```

In [16]:

```python
temp = pd.read_table("slump_test.data", sep=",")
temp2 = pd.read_table("slump_test.names")
display(temp)
#drop the No column along with the target variables other than 280day Compressive Stren
gth
temp = temp.drop(columns=["No", "SLUMP(cm)", "FLOW(cm)"])
temp.head()
```

	No	Cement	Slag	Fly ash	Water	SP	Coarse Aggr.	Fine Aggr.	SLUMP(cm)	FLOW(cm)	Compr Streng day
0	1	273.0	82.0	105.0	210.0	9.0	904.0	680.0	23.0	62.0	
1	2	163.0	149.0	191.0	180.0	12.0	843.0	746.0	0.0	20.0	
2	3	162.0	148.0	191.0	179.0	16.0	840.0	743.0	1.0	20.0	
3	4	162.0	148.0	190.0	179.0	19.0	838.0	741.0	3.0	21.5	
4	5	154.0	112.0	144.0	220.0	10.0	923.0	658.0	20.0	64.0	
...	...	...	...	...	...	...	...	...	...	...	
98	99	248.3	101.0	239.1	168.9	7.7	954.2	640.6	0.0	20.0	
99	100	248.0	101.0	239.9	169.1	7.7	949.9	644.1	2.0	20.0	
100	101	258.8	88.0	239.6	175.3	7.6	938.9	646.0	0.0	20.0	
101	102	297.1	40.9	239.9	194.0	7.5	908.9	651.8	27.5	67.0	
102	103	348.7	0.1	223.1	208.5	9.6	786.2	758.1	29.0	78.0	

103 rows × 11 columns

Out[16]:

	Cement	Slag	Fly ash	Water	SP	Coarse Aggr.	Fine Aggr.	Compressive Strength (28-day) (Mpa)
0	273.0	82.0	105.0	210.0	9.0	904.0	680.0	34.99
1	163.0	149.0	191.0	180.0	12.0	843.0	746.0	41.14
2	162.0	148.0	191.0	179.0	16.0	840.0	743.0	41.81
3	162.0	148.0	190.0	179.0	19.0	838.0	741.0	42.08
4	154.0	112.0	144.0	220.0	10.0	923.0	658.0	26.82

Basic EDA

In [17]:

```
temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 8 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Cement                                   103 non-null    float64
 1   Slag                                     103 non-null    float64
 2   Fly ash                                  103 non-null    float64
 3   Water                                    103 non-null    float64
 4   SP                                       103 non-null    float64
 5   Coarse Aggr.                             103 non-null    float64
 6   Fine Aggr.                               103 non-null    float64
 7   Compressive Strength (28-day)(Mpa)       103 non-null    float64
dtypes: float64(8)
memory usage: 6.6 KB
```

In [18]:

```
temp.describe().transpose()
```

Out[18]:

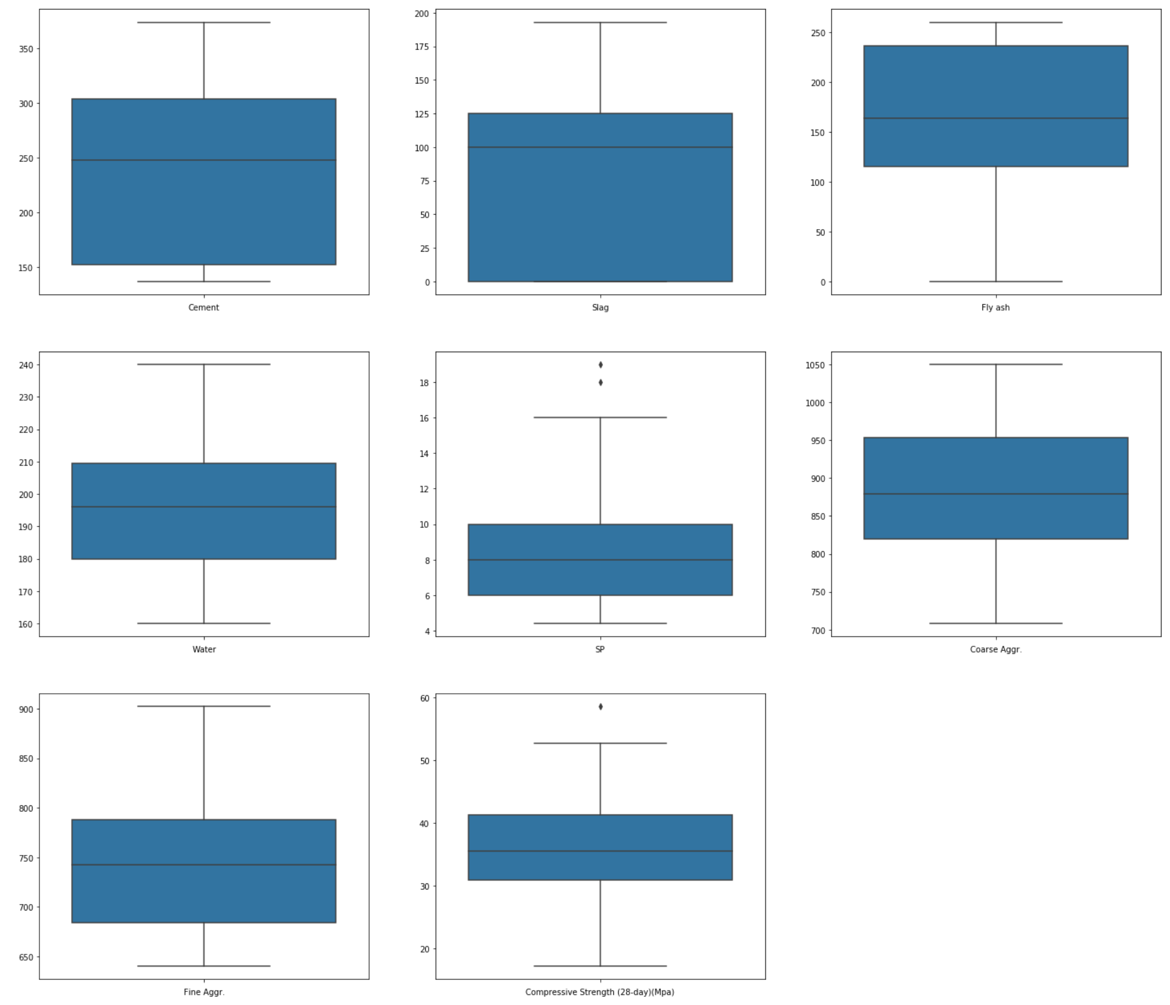
	count	mean	std	min	25%	50%	75%	max
Cement	103.0	229.894175	78.877230	137.00	152.00	248.00	303.900	374.00
Slag	103.0	77.973786	60.461363	0.00	0.05	100.00	125.000	193.00
Fly ash	103.0	149.014563	85.418080	0.00	115.50	164.00	235.950	260.00
Water	103.0	197.167961	20.208158	160.00	180.00	196.00	209.500	240.00
SP	103.0	8.539806	2.807530	4.40	6.00	8.00	10.000	19.00
Coarse Aggr.	103.0	883.978641	88.391393	708.00	819.50	879.00	952.800	1049.90
Fine Aggr.	103.0	739.604854	63.342117	640.60	684.50	742.70	788.000	902.00
Compressive Strength (28-day) (Mpa)	103.0	36.039417	7.838232	17.19	30.90	35.52	41.205	58.53

Observations:

1. This dataset has no null values
2. The data is not scaled and if we were to perform PCA on the dataset then we would have to scale and/or standardize the data
3. Columns like Cement, Slag, Fly ash, Water are somewhat evenly distributed with few outliers since the 75% mark is close to the maximum value of the column. However, we can confirm this using boxplots.

In [19]:

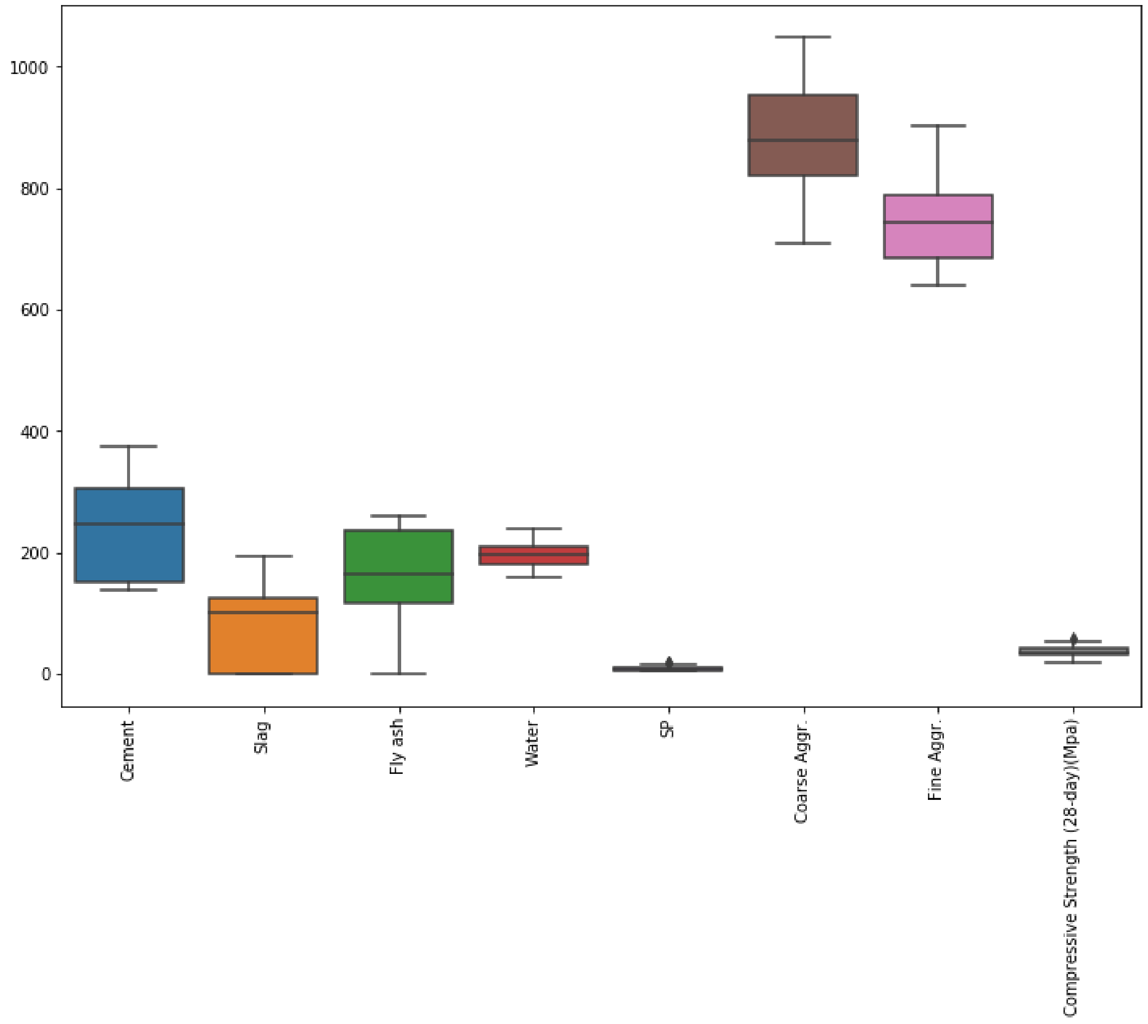
```
#Confirming the distributions
#names of the columns
col_names = temp.columns
col_names

#plotting boxplots
plt.figure(figsize=(25,30))
for i in range(len(col_names)):
  plt.subplot(4,3,i+1)
  sns.boxplot(y=temp[col_names[i]])
  plt.xlabel(col_names[i])
  plt.ylabel("")
```



As expected, most columns, except SP and the target variable are evenly distributed.

In [20]:

```
#We can also plot them on a single plot since they all have the same units (kg of compo
nent in 1 M^3 of concrete )
plt.subplots(figsize=(12, 8))
ax = sns.boxplot(data=temp)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90);
```

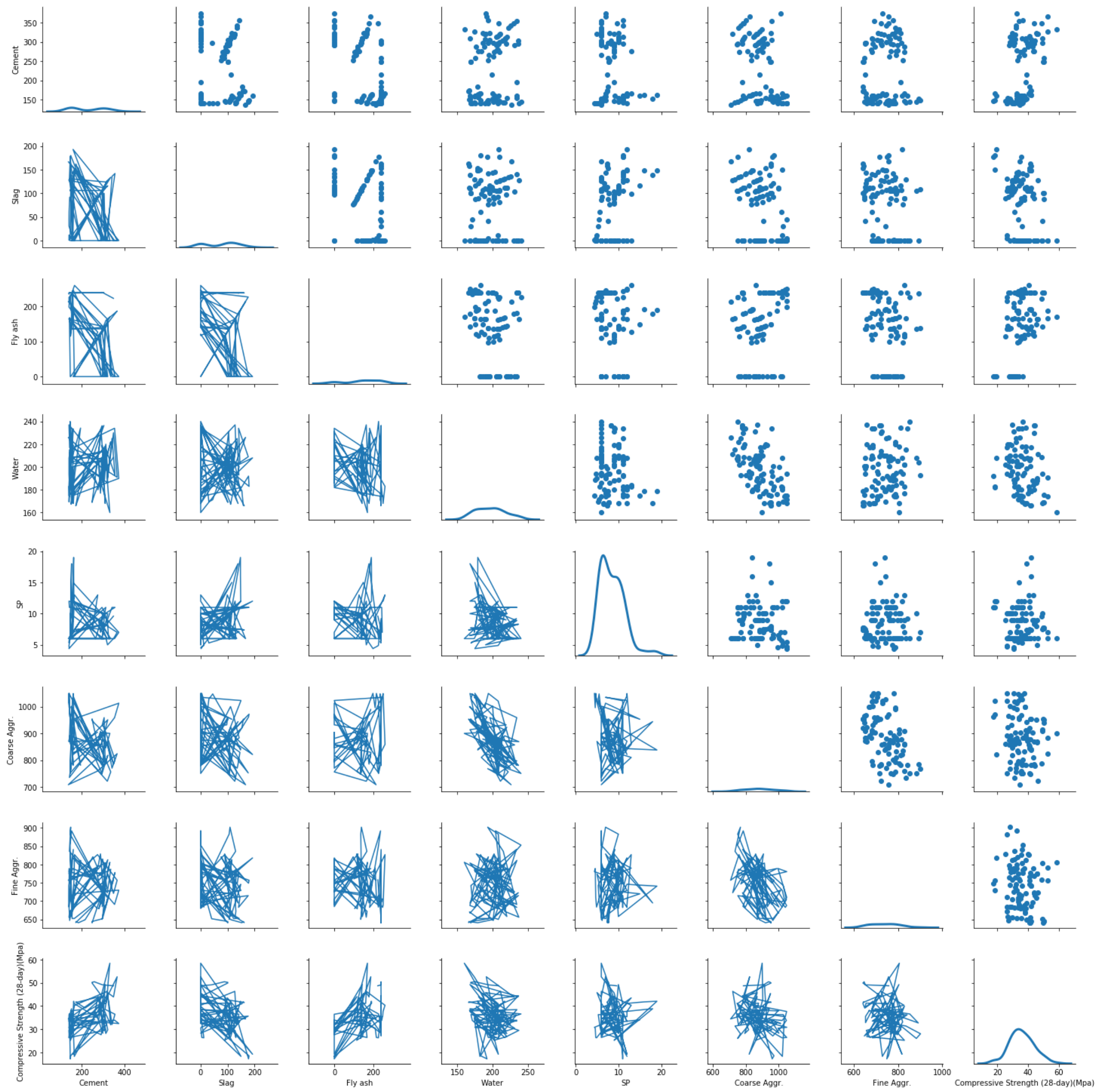


In [21]:

```
#Understanding the correlation between the variables
r = sns.PairGrid(temp)
r.map_upper(plt.scatter)
r.map_diag(sns.kdeplot, lw=3, legend=True)
r.map_lower(plt.plot)
```

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x1aa9cc3d108>
```



Observation:

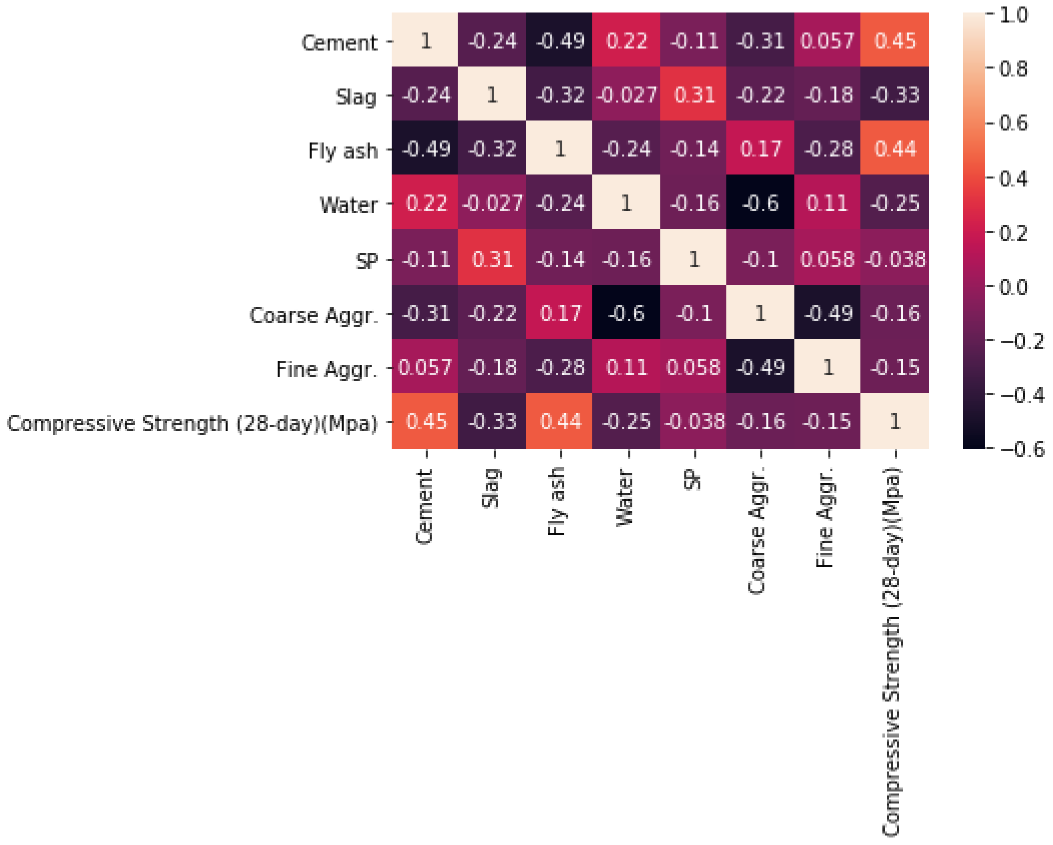
1. Cement, Slag, Fly Ash all seem to have 2 peaks hence they have a bimodal distribution which would make sense intuitively.
2. SP also has 2 peaks but one of them is higher than the other one.
3. Cement and Fly Ash seem to have a positive correlation with compressive strength (our target variable)
4. Water and Coarse Aggr. have a weak negative correlation

In [22]:

```
#Plotting Heat Maps to get a more accurate
sns.heatmap(temp.corr(), annot = True)
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa9d6c1408>
```



	Cement	Slag	Fly ash	Water	SP	Coarse Aggr.	Fine Aggr.	Compressive Strength (28-day)(Mpa)
Cement	1	-0.24	-0.49	0.22	-0.11	-0.31	0.057	0.45
Slag	-0.24	1	-0.32	-0.027	0.31	-0.22	-0.18	-0.33
Fly ash	-0.49	-0.32	1	-0.24	-0.14	0.17	-0.28	0.44
Water	0.22	-0.027	-0.24	1	-0.16	-0.6	0.11	-0.25
SP	-0.11	0.31	-0.14	-0.16	1	-0.1	0.058	-0.038
Coarse Aggr.	-0.31	-0.22	0.17	-0.6	-0.1	1	-0.49	-0.16
Fine Aggr.	0.057	-0.18	-0.28	0.11	0.058	-0.49	1	-0.15
Compressive Strength (28-day)(Mpa)	0.45	-0.33	0.44	-0.25	-0.038	-0.16	-0.15	1

Observations:

1. As we predicted, Cement and Fly Ash have a positive correlation to Compressive Strength while we learn that Slag and Water are negatively correlated to Compressive Strength. Sp, Coarse Aggr, and Fine Aggr. are all weakly correlated to Compressive Strength.
2. Cement and Fly-ash have a strong negative correlation
3. Water and Coarse Aggr. have a strong negative correlation.

In [ ]:

PCA

While we have made some predictions, I would like to try another method to see if we can learn something more about the dataset

In [23]:

```
temp.columns
```

Out[23]:

```
Index(['Cement', 'Slag', 'Fly ash', 'Water', 'SP', 'Coarse Aggr.',
       'Fine Aggr.', 'Compressive Strength (28-day)(Mpa)'],
      dtype='object')
```

In [24]:

```
#Step 1: Extract the features from the dataset
data = temp.drop(columns=('Compressive Strength (28-day)(Mpa)'), inplace=False)

#Step 2: Standardize the data
x = StandardScaler().fit_transform(X = data.values)
display(x.shape, np.mean(x), np.std(x))

#convert the normalized features back into a dataframe
feature_cols = data.columns
concrete_norml = pd.DataFrame(data=x, columns=feature_cols)
concrete_norml.tail()
```

```
(103, 7)

-2.0572232467395412e-16

1.0
```

Out[24]:

	Cement	Slag	Fly ash	Water	SP	Coarse Aggr.	Fine Aggr.
98	0.234489	0.382704	1.059799	-1.405679	-0.300589	0.798321	-1.570661
99	0.230667	0.382704	1.069210	-1.395734	-0.300589	0.749436	-1.515135
100	0.368258	0.166639	1.065681	-1.087427	-0.336382	0.624381	-1.484993
101	0.856197	-0.616180	1.069210	-0.157533	-0.372174	0.283322	-1.392979
102	1.513577	-1.294291	0.871569	0.563508	0.379472	-1.111610	0.293416

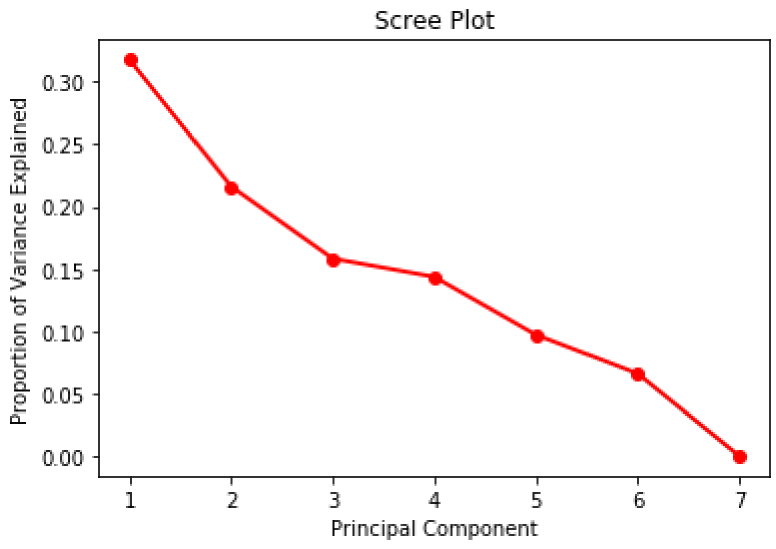
In [25]:

```
#using the sklearn library to perform pca on the normalised dataset

pca_concrete = PCA(n_components=7)
principalComp_concrete = pca_concrete.fit_transform(x)

print('The proportion of variance explained by each PC: {}'.format(pca_concrete.explain
ed_variance_ratio_))
```

```
The proportion of variance explained by each PC: [0.31779551 0.21620366 0.
15837992 0.14373075 0.09732565 0.0661284
 0.00043611]
```

In [26]:

```
#creating a scree plot to determine th number of components to be used
PC_values = np.arange(pca_concrete.n_components_) + 1
plt.plot(PC_values, pca_concrete.explained_variance_ratio_, 'ro-', linewidth=2)
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Proportion of Variance Explained')
plt.show()
```



Ideally, PCA is performed on datasets with a large number of columns, which might mean that a big proportion of them are irrelevant. In this case, we only have 7 columns out of which 4 are the key components that make up concrete:

1. Water
2. Coarse aggr.
3. Fine aggr.
4. Cement

However, we can see in the scree plot that the first 4 PCs explain 83.6% of the variance. We can make an educated guess that the part discarded could be a combination of the additional chemicals added to concrete such as superplasticizer, fly ash which we saw had a positive correlation with strength of the cement and could contribute to it durability.

In [27]:

```
sum(pca_concrete.explained_variance_ratio_[:4])
```

Out[27]:

```
0.8361098405885852
```

In [28]:

```
#creating a dataframe with the pca components of the all the samples
pc_concrete_df = pd.DataFrame(data=principalComp_concrete, columns=['pc_'+str(i) for i
in range(1,8)])
pc_concrete_df = pc_concrete_df.drop(columns=["pc_5","pc_6","pc_7"])
pc_concrete_df
```

Out[28]:

	pc_1	pc_2	pc_3	pc_4
0	0.267519	0.048998	-0.944112	0.746884
1	-0.488953	1.844713	0.832270	-0.288779
2	-0.452798	2.699056	0.902533	-0.783674
3	-0.398914	3.341436	0.950733	-1.136015
4	-0.490634	0.849283	0.053376	1.469176
...	...	...	...	...
98	-2.013182	0.087107	-0.849752	0.523874
99	-1.966723	0.083172	-0.798182	0.502728
100	-1.710631	-0.174662	-0.748766	0.584401
101	-0.943042	-1.026062	-0.638782	0.746259
102	1.099560	-1.353299	0.336996	-0.344661

103 rows × 4 columns

This dataframe can be further used for regression. (PC Regression)

In [ ]:

```

```

In [ ]:

```

```