**Assignment**

**Dataset Link**

<https://drive.google.com/file/d/1pP0Rr83ri0voscgr95-YnVCBv6BYV22w/view>

**Problem 1:**

There are various stocks for which we have collected a data set, which all stocks are apparently similar in performance

**Problem 2:**

How many Unique patterns that exist in the historical stock data set, based on fluctuations in price.

**Problem 3:**

Identify which all stocks are moving together and which all stocks are different from each other.

**Task:**Deploy this assignment in any cloud platform.(Try to look for free cloud platform)

**Assignment:** Submit assignment’s deployable link only.

**Importing all Libraries**

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.cluster **import** KMeans

**from** sklearn.decomposition **import** PCA

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.cluster **import** DBSCAN

**from** sklearn **import** metrics

**%matplotlib** inline

df **=** pd**.**read\_csv('data\_stocks.csv')

print(df**.**shape)

df**.**head()

**del** df['DATE']

df**.**head()

df**.**describe()

print(f'Number of Share Name:{len(df**.**columns)}')

print('Number of null values:',df**.**isnull()**.**sum()**.**sum())

df1 **=** df**.**T**.**copy()

df1**.**head()

len(df1**.**index)

df1**.**shape

**DBSCAN**

X **=** StandardScaler()**.**fit\_transform(df1)

X**.**shape

db **=** DBSCAN(eps**=**1.6, min\_samples**=**5)**.**fit(X)

labels **=** db**.**labels\_

data **=** pd**.**DataFrame(X)

data**.**head()

data['Class'] **=** labels

data['Class']**.**unique()

df1['CLASS'] **=** labels

## Problem 1:

Similar\_share1 **=** df1[df1['CLASS']**==**0]

Similar\_share2 **=** df1[df1['CLASS']**==**1]

Disimilar\_share **=** df1[df1['CLASS']**==-**1]

print(f'Number of Similar Stocks are:{len(Similar\_share1)} and {len(Similar\_share2)}')

print(f'Number of UnSimilar Stocks are:{len(Disimilar\_share)}')

print('Similar Stoks in Class One Are:-')

**for** stoks1 **in** Similar\_share1**.**index:

print(stoks1)

print('Similar Stoks in Class Two Are:-')

**for** stoks2 **in** Similar\_share2**.**index:

print(stoks2)

print(f'Others :{len(Disimilar\_share)} Stocks are Disimilar Stocks')

print('The Stocks which are not Similar to each other are:-\n')

**for** stoks3 **in** Disimilar\_share**.**index:

print(stoks3,end**=**' ')

## Graphs of similar Stocks in Two Different Classes

Stocks1 **=** df[Similar\_share1**.**index]

Stocks1**.**plot(figsize**=**(20,6))

plt**.**title('Similar Stocks One')

plt**.**show()

Stocks2 **=** df[Similar\_share2**.**index]

Stocks2**.**plot(figsize**=**(20,6))

plt**.**title('Similar Stocks Two')

plt**.**show()

## Problem 2:

print(f"The number of Unique patterns that exist in the historical stock data set, based on fluctuations in price is: {len(df1['CLASS']**.**unique())}")

## Problem 3:

print('The Stocks from class One which are Moving Together:')

**for** stoks1 **in** Similar\_share1**.**index:

print(stoks1)

print('The Stocks from class Two which are Moving Together:')

**for** stoks2 **in** Similar\_share2**.**index:

print(stoks2)

print('The Stocks which are Different:\n\n',list(Disimilar\_share**.**index))

**del** d['Class']

## Making Clustering with k-Means with compressed data (2 Principal Components)

pca **=** PCA(n\_components**=**2,random\_state**=**0)

scaler **=** StandardScaler()

X **=** scaler**.**fit\_transform(d)

pca**.**fit(X)

X **=** pca**.**transform(X)

X **=** pd**.**DataFrame(X,columns**=**['1stPrinc','2ndPrinc'])

*##Elbow Graph*

wcss**=**[]

**for** i **in** range (1,11):

kmeans**=**KMeans(n\_clusters**=**i,init**=**'k-means++',random\_state**=**0)

kmeans**.**fit(X)

wcss**.**append(kmeans**.**inertia\_)

plt**.**plot(range(1,11),wcss)

plt**.**title('The Elbow Method')

plt**.**xlabel('Number of clusters')

plt**.**ylabel('WCSS')

plt**.**show()

## Clustering with Compressed Data

kmeans **=** KMeans(n\_clusters **=** 4, init **=** 'k-means++',max\_iter**=**1000, random\_state **=** 0)

y\_kmeans **=** kmeans**.**fit\_predict(X)

centers **=** kmeans**.**cluster\_centers\_

centrex **=** centers[:,0]

centrey **=** centers[:,1]

X['Class'] **=** y\_kmeans

X**.**head()

X['Class']**.**unique()

array([1, 0, 3, 2], dtype=int32)

plt**.**figure(figsize**=**(8,6))

sns**.**scatterplot(X['1stPrinc'],X['2ndPrinc'],hue**=**X['Class'],s**=**200,markers**=**'o',palette**=**'tab10')

plt**.**scatter(centrex,centrey,color**=**'k',marker**=**'o',linewidths**=**5,label**=**'centroids')

plt**.**title('Cluster from Compressed Data sets')

plt**.**xlabel('1st Principal Component')

plt**.**ylabel('2nd Principal Component')

plt**.**legend()

plt**.**show()

d['Class'] **=** y\_kmeans

zero **=** d[d['Class']**==**0]

one **=** d[d['Class']**==**1]

two **=** d[d['Class']**==**2]

three **=** d[d['Class']**==**3]

print(f'Without removing Noise in zero class number of stocks are :{len(zero**.**index)}')

print(f'Without removing Noise in one class number of stocks are :{len(one**.**index)}')

print(f'Without removing Noise in two class number of stocks are :{len(two**.**index)}')

print(f'Without removing Noise in three class number of stocks are :{len(three**.**index)}')

print('Stocks in Class Zero are:\n')

**for** stoks0 **in** zero**.**index:

print(stoks0,end**=**' ')

print('Stocks in Class one are:\n')

**for** stoks1 **in** one**.**index:

print(stoks1)

print('Stocks in Class Two are:\n')

**for** stoks2 **in** two**.**index:

print(stoks2)

print('Stocks in Class Three are:\n')

**for** stoks3 **in** three**.**index:

print(stoks3,end**=**' ')

**Similar Stocks of zero Class extracted from compressed data sets (including noises)**

df[zero**.**index]**.**plot(figsize**=**(20,40))

plt**.**show()

**Similar Stocks of one Class extracted from compressed data sets**

df[one**.**index]**.**plot(figsize**=**(20,6))

plt**.**show()

**Similar Stocks of two Class extracted from compressed data sets**

df[two**.**index]**.**plot(figsize**=**(20,6))

plt**.**show()

**Similar Stocks of Three Class extracted from compressed data sets**

df[three**.**index]**.**plot(figsize**=**(20,20))

plt**.**show()