# Case Study 2 : Clustering of drivers on the basis of Speed and Distance they covered.

In driver\_data.csv you are given a dataset of driver.

For each driver we have two features: mean distance driven per day and the mean percentage of time a driver was >5 mph over the speed limit.

Run the K-Means algorithm on this dataset and find the optimal number of clusters. What insights can you gather from these clusters?

First import the dataset

In [1]:

import pandas as pd

# 1) Acquire the Data

In [2]:

df = pd.read\_csv('driver\_data.csv')

In [3]:

df.head()

Out[3]:

|  |  |  |
| --- | --- | --- |
|  | **Distance\_Feature** | **Speeding\_Feature** |
| **0** | 71.24 | 28 |
| **1** | 52.53 | 25 |
| **2** | 64.54 | 27 |
| **3** | 55.69 | 22 |
| **4** | 54.58 | 25 |

Scaling Input Features using Standardization where (mean=0 and std =1)

# 2) Preprocess the Data

In [4]:

from sklearn.preprocessing import StandardScaler

In [5]:

sc = StandardScaler()

In [6]:

df\_scaled = sc.fit\_transform(df)

D:\anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645: DataConversionWarning: Data with input dtype int64, float64 were all converted to float64 by StandardScaler.

return self.partial\_fit(X, y)

D:\anaconda3\lib\site-packages\sklearn\base.py:464: DataConversionWarning: Data with input dtype int64, float64 were all converted to float64 by StandardScaler.

return self.fit(X, \*\*fit\_params).transform(X)

In [7]:

df\_scaled[:5,]

Out[7]:

array([[-0.0898104 , 1.26061251],

[-0.43977285, 1.04174351],

[-0.215131 , 1.18765617],

[-0.38066642, 0.8228745 ],

[-0.40142849, 1.04174351]])

Runing multiple iteration of KMeans algorithm to find an optimal number of clusters using elbow method

# 3) Train the Model

In [8]:

from sklearn.cluster import KMeans

In [9]:

ssq = []

for K in range(1,11):

my\_model = KMeans(n\_clusters=K, random\_state=123)

result = my\_model.fit(df\_scaled)

ssq.append(result.inertia\_)

Generate Scree plot

In [10]:

import matplotlib.pyplot as plt

%matplotlib inline

In [11]:

plt.plot(range(1,11), ssq, marker='o')

plt.xlabel("Number of clusters")

plt.ylabel("Within-cluster SSQ")

plt.title("Scree Plot")

plt.plot([4]\*3000, range(1,3001), ":")

plt.text(4.1, 3000, "optimal number of clusters = 4")

plt.show()

From the result we see that 4 should be the optimal number of clusters for the given data

Let's fit the KMeans model again with K=4

In [12]:

my\_model = KMeans(n\_clusters=4, random\_state=163)

In [13]:

result = my\_model.fit(df\_scaled)

In [14]:

result.labels\_

Out[14]:

array([2, 2, 2, ..., 0, 0, 0])

Make predictions for the clusters of the given dataset

# 4) Test the Model

In [15]:

predictions = result.predict(df\_scaled)

In [16]:

predictions[:5]

Out[16]:

array([2, 2, 2, 2, 2])

Plot the data partitioned into clusters

In [17]:

plt.scatter(df\_scaled[predictions==0,0], df\_scaled[predictions==0, 1], s=50, c='lightgreen',\

marker='s', edgecolors='black', label='cluster 1')

plt.scatter(df\_scaled[predictions==1,0], df\_scaled[predictions==1, 1], s=50, c='orange',\

marker='o', edgecolors='black', label='cluster 2')

plt.scatter(df\_scaled[predictions==2,0], df\_scaled[predictions==2, 1], s=50, c='lightblue',\

marker='v', edgecolors='black', label='cluster 3')

plt.scatter(df\_scaled[predictions==3,0], df\_scaled[predictions==3,1],\

s=50, c='yellow', marker='s',edgecolors='black',label='cluster 4')

plt.scatter(result.cluster\_centers\_[:,0], result.cluster\_centers\_[:,1], s=250, c='red',\

marker='\*', edgecolors='black', label='centroids')

plt.legend(scatterpoints=1)

plt.xlabel("Driving Distance")

plt.ylabel("Driving Speed ")

plt.title("Clustering Output")

plt.show()

Cluster 1(LightGreen): Group represents higher driving distance with lower speeding score.

Cluster 2(Orange): Group represents lower driving distance with lower speeding score.

Cluster 3(LightBlue): Group represents lower driving distance with moderate speeding score.

Cluster 4(Yellow): Group represents higher driving distance with higher driving speed.

Accordingly, an analyst may conclude following labels for each cluster:

Cluster 1(LightGreen): A Cluster 2(Orange): B Cluster 3(LightBlue): C Cluster 4(Yellow): D

In [18]:

import numpy as np

In [19]:

predictions\_relabelled = np.where(predictions==0, "A", np.where(predictions==1,\

"B", np.where(predictions==2,"C","D")))

df['category'] = pd.Series(predictions\_relabelled, index=df.index)

df.index.name = "Number"

In [20]:

pd.DataFrame(df).head()

Out[20]:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Distance\_Feature** | **Speeding\_Feature** | **category** |
| **Number** |  |  |  |
| **0** | 71.24 | 28 | C |
| **1** | 52.53 | 25 | C |
| **2** | 64.54 | 27 | C |
| **3** | 55.69 | 22 | C |
| **4** | 54.58 | 25 | C |

In [22]:

pd.DataFrame(df).tail()

Out[22]:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Distance\_Feature** | **Speeding\_Feature** | **category** |
| **Number** |  |  |  |
| **3995** | 160.04 | 10 | A |
| **3996** | 176.17 | 5 | A |
| **3997** | 170.91 | 12 | A |
| **3998** | 176.14 | 5 | A |
| **3999** | 168.03 | 9 | A |

In [21]:

df.to\_csv("driver\_segementation\_output.csv", index=False)