Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

Bhuvi Ghosh
60009210191

AY: 2022-23

Experiment 9

(K-Means)

Aim: Explore K means clustering with variations on different datasets.

Theory:

The K-means clustering algorithm computes centroids and repeats until the optimal centroid is found. It is presumptively known how many clusters there are. It is also known as the flat clustering algorithm. The number of clusters found from data by the method is denoted by the letter 'K' in K- means.

In this method, data points are assigned to clusters in such a way that the sum of the squared distances between the data points and the centroid is as small as possible. It is essential to note that reduced diversity within clusters leads to more identical data points within the same cluster.

The following stages will help us understand how the K-Means clustering technique works-

Step 1: First, we need to provide the number of clusters, K, that need to be generated by this algorithm.

Step 2: Next, choose K data points at random and assign each to a cluster. Briefly, categorize the data based on the number of data points.

Step 3: The cluster centroids will now be computed.

Step 4: Iterate the steps below until we find the ideal centroid, which is the assigning of data points to clusters that do not vary.

4.1 The sum of squared distances between data points and centroids would be calculated first.

4.2 At this point, we need to allocate each data point to the cluster that is closest to the others (centroid).

4.3 Finally, compute the centroids for the clusters by averaging all of the cluster's data points.

When using the K-means algorithm, we must keep the following points in mind:

It is suggested to normalize the data while dealing with clustering algorithms such as K-Means since such algorithms employ distance-based measurement to identify the similarity between data points.

Because of the iterative nature of K-Means and the random initialization of centroids, K-Means may become stuck in a local optimum and fail to converge to the global optimum. As a result, it is advised to employ distinct centroids' initializations.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:

Dataset 1: Synthetic Data (200 samples, 3 clusters and cluster_std = 2.7)

Dataset 2: Titanic dataset
(http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv

Task 1: Perform Kmeans clustering on Dataset 1 with random initialisation, 10 variations of initial means, 300 iteration. Find Lowest SSE value, final location of centroids and number of iterations to converge. Show the predicted labels for first 10 points.

Task 1

```
[1] from sklearn.datasets import make_blobs

    from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_score
    from sklearn.preprocessing import StandardScaler
```

```
[2] features, true_labels = make_blobs(n_samples=200, centers=3,cluster_std=2.75, random_state=0)
```

```
[3] features[:5]
```

```
    array([[-2.09588912,  5.33837427],
           [ 0.01951167,  4.73374699],
           [ 1.4023755 ,  5.34373426],
           [-1.25816035,  4.52100488],
           [ 1.79642506,  7.95134854]])
```

```
[4] true_labels[:5]
```

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

```
[5] scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)
```

```
[6] scaled_features[:5]
```

```
    array([[-0.73411211,  0.82841827],
           [-0.0659294 ,  0.63727076],
           [ 0.37086995,  0.83011278],
           [-0.46950226,  0.57001424],
           [ 0.49533672,  1.65448674]])
```

```
[7] kmeans = KMeans(init = "random", n_clusters=3, n_init=10, max_iter=300, random_state=42)
```

```
[8] kmeans.fit(scaled_features)
```

```
KMeans(init='random', n_clusters=3, n_init=10, random_state=42)
```

```
[9] kmeans.inertia_
```

```
180.33313645221375
```

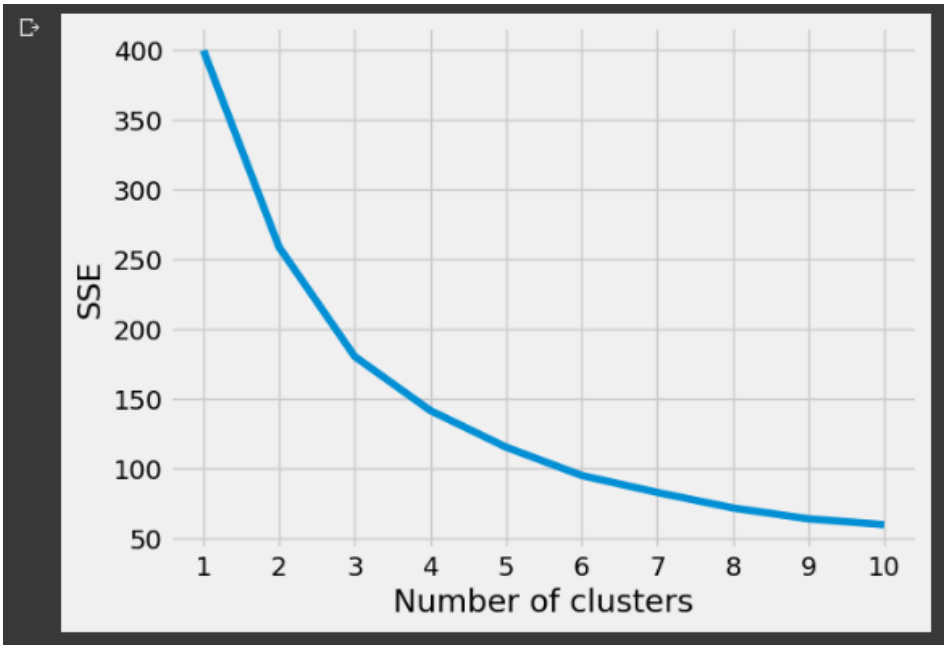
```
[10] kmeans.cluster_centers_
```

```
array([[ 1.13610094, -0.45307413],
       [-0.20826956,  1.09470895],
       [-0.61185038, -0.57796256]])
```

```
[11] kmeans_kwargs = {'init':'random','n_init':10,'max_iter':300,'random_state':42}
```

```
[12] sse = []
     for k in range(1,11):
       kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
       kmeans.fit(scaled_features)
       sse.append(kmeans.inertia_)
```

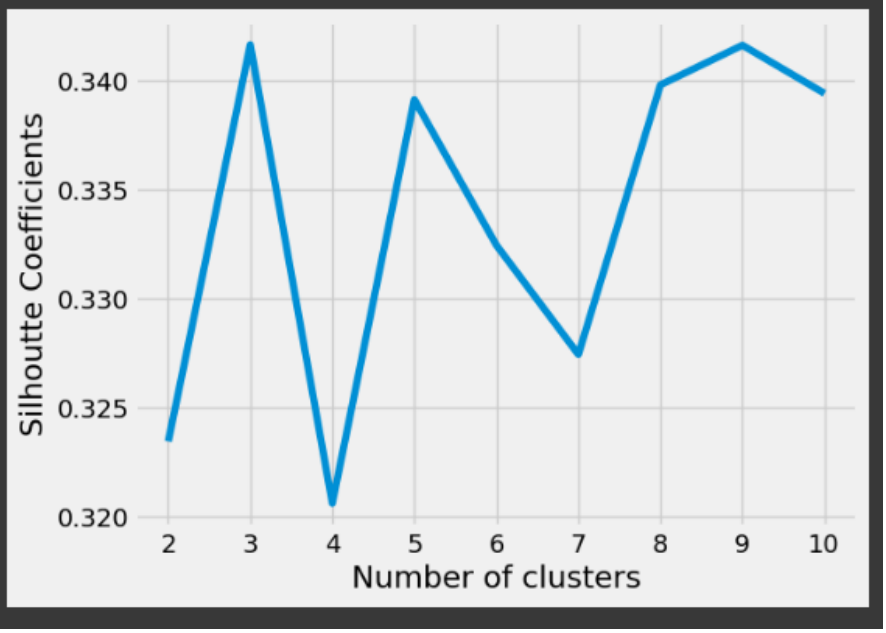
```
import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')
plt.plot(range(1,11),sse)
plt.xticks(range(1,11))
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```



Task 2: Perform elbow method and silhouette method to find appropriate clustering value on Dataset 1.

```
[14] silhouette_coefficients = []
     for k in range(2,11):
       kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
       kmeans.fit(scaled_features)
       score = silhouette_score(scaled_features, kmeans.labels_)
       silhouette_coefficients.append(score)
```

```
plt.style.use('fivethirtyeight')
plt.plot(range(2,11),silhouette_coefficients)
plt.xticks(range(2,11))
plt.xlabel('Number of clusters')
plt.ylabel('Silhoutte Coefficients')
plt.show()
```



Task 3: Preform data cleaning and pre-processing on dataset 2. Form three clustering using Kmeans++ initialisation.

```
[ ] import tarfile
    import urllib
    import pandas as pd
    import seaborn as sns

    from sklearn.decomposition import PCA
    from sklearn.metrics import silhouette_score, adjusted_rand_score
    from sklearn.pipeline import Pipeline
    from sklearn.preprocessing import LabelEncoder, MinMaxScaler
```

```
[ ] train_data = 'http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv'
    train_data = pd.read_csv(train_data)
    test_data = 'http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv'
    test_data = pd.read_csv(test_data)
```

```
[ ] train_data.head(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
[ ] # DROPPING UNWANTED COLUMNS
    train_data.drop('Name', axis=1, inplace=True)
    train_data.drop('Ticket', axis=1, inplace=True)
    train_data.drop('Cabin', axis=1, inplace=True)

    # FILLING MISSING AGE VALUE BY MEDIAN
    train_data.fillna(value=train_data['Age'].median(), inplace=True)
```

```
    # DROPPING UNWANTED COLUMNS
    test_data.drop('Name', axis=1, inplace=True)
    test_data.drop('Ticket', axis=1, inplace=True)
    test_data.drop('Cabin', axis=1, inplace=True)
```

```
[ ] # FILLING MISSING AGE VALUE BY MEDIAN
    test_data.fillna(value=test_data['Age'].median(), inplace=True)
```

```
[ ] test_data.isnull().sum()
```

```
PassengerId    0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

```
[ ] train_data.dropna(inplace=True)
```

```
[ ] train_data.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

```
[ ] test_data.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

```
[ ] test_data.dropna(inplace=True)
```

```
train_data.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

```
test_data.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

```
test_data.dropna(inplace=True)
```

```
test_data.isnull().sum()
```

```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
le = LabelEncoder()

train_data['Sex'] = le.fit_transform(train_data['Sex'])
train_data['Embarked'] = le.fit_transform(train_data['Embarked'].astype(str))

test_data['Sex'] = le.fit_transform(test_data['Sex'])
test_data['Embarked'] = le.fit_transform(test_data['Embarked'].astype(str))
```

```
[ ] # GETTING TRAINING FEATURES AND LABELS
train_features = train_data.iloc[:,:-1].values
train_label = train_data.iloc[:,-1].values.reshape(-1,1)
train_features
```

```
array([[  1.    ,   0.    ,   3.    , ...,   1.    ,   0.    ,   7.25  ],
       [  2.    ,   1.    ,   1.    , ...,   1.    ,   0.    ,  71.2833],
       [  3.    ,   1.    ,   3.    , ...,   0.    ,   0.    ,   7.925 ],
       ...,
       [889.    ,   0.    ,   3.    , ...,   1.    ,   2.    ,  23.45  ],
       [890.    ,   1.    ,   1.    , ...,   0.    ,   0.    ,  30.    ],
       [891.    ,   0.    ,   3.    , ...,   0.    ,   0.    ,   7.75  ]])
```

```
[ ] # GETTING TRAINING FEATURES AND LABELS
test_features = test_data.iloc[:,:-1].values
test_label = test_data.iloc[:,-1].values.reshape(-1,1)
```

```
[ ] from sklearn.cluster import KMeans
```

```
[ ] # CREATING 3 CLUSTERS WITH INIT RANDOM
    kmeans = KMeans(init="k-means++",n_clusters=3,n_init=10,max_iter=300,random_state=42)
```

```
[ ] # FITTING THE SCALED FEATURES
    kmeans.fit(train_features)
```

```
▾ KMeans
KMeans(n_clusters=3, n_init=10, random_state=42)
```

```
# GETTING POSTION OF CENTRIODS OF CLUSTERS
centers=kmeans.cluster_centers_
print(centers)
```

```
[[1.49000000e+02 3.56902357e-01 2.38720539e+00 6.46464646e-01
  2.87923569e+01 6.16161616e-01 3.80471380e-01 2.85651364e+01]
 [4.46500000e+02 4.29530201e-01 2.22818792e+00 6.07382550e-01
  2.98780872e+01 4.63087248e-01 3.72483221e-01 3.52431903e+01]
 [7.43500000e+02 3.64864865e-01 2.31081081e+00 6.89189189e-01
  2.94127365e+01 4.89864865e-01 3.91891892e-01 3.27960578e+01]]
```

```
[ ] # FINDING LOWEST SSE
    print("LOWEST SSE OF ALL 3 CLUSTERS IS:", kmeans.inertia_)
```

```
LOWEST SSE OF ALL 3 CLUSTERS IS: 8893946.462558528
```

```
[ ] # NUMBER OF ITERATIONS
    kmeans.n_iter_
```

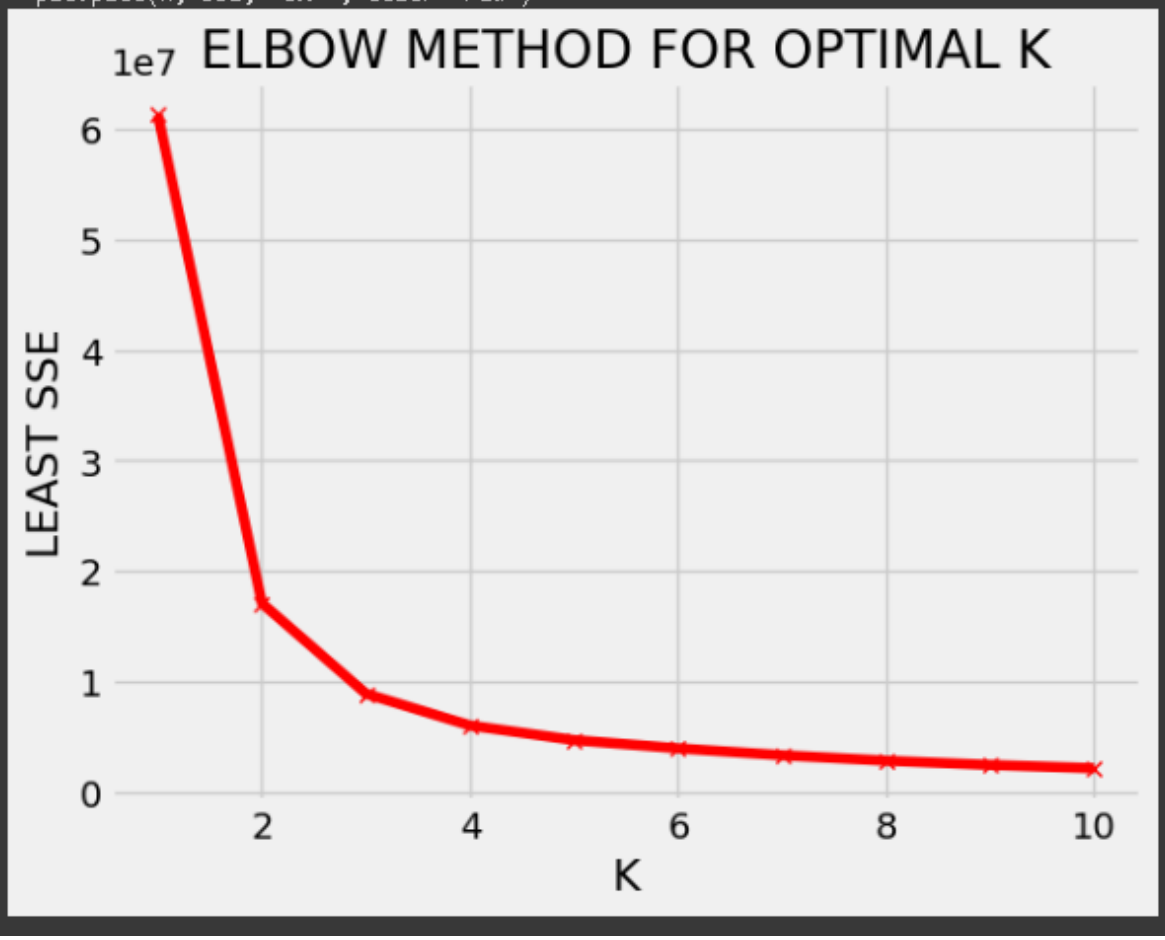
```
12
```

```
[ ] SSE = []
    K = range(1,11)
    for k in K:
        km = KMeans(n_clusters=k)
        km = km.fit(train_features)
        SSE.append(km.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

```
[ ] #VISUALIZING PLOT
    plt.style.use("fivethirtyeight")
    plt.plot(K, SSE, 'bx-', color='red')
    plt.xlabel('K')
    plt.ylabel('LEAST SSE')
    plt.title('ELBOW METHOD FOR OPTIMAL K')
    plt.show()
```

```
<ipython-input-179-26834c19a141>:3: UserWarning: color is redundantly defined by the
  plt.plot(K, SSE, 'bx-', color='red')
```

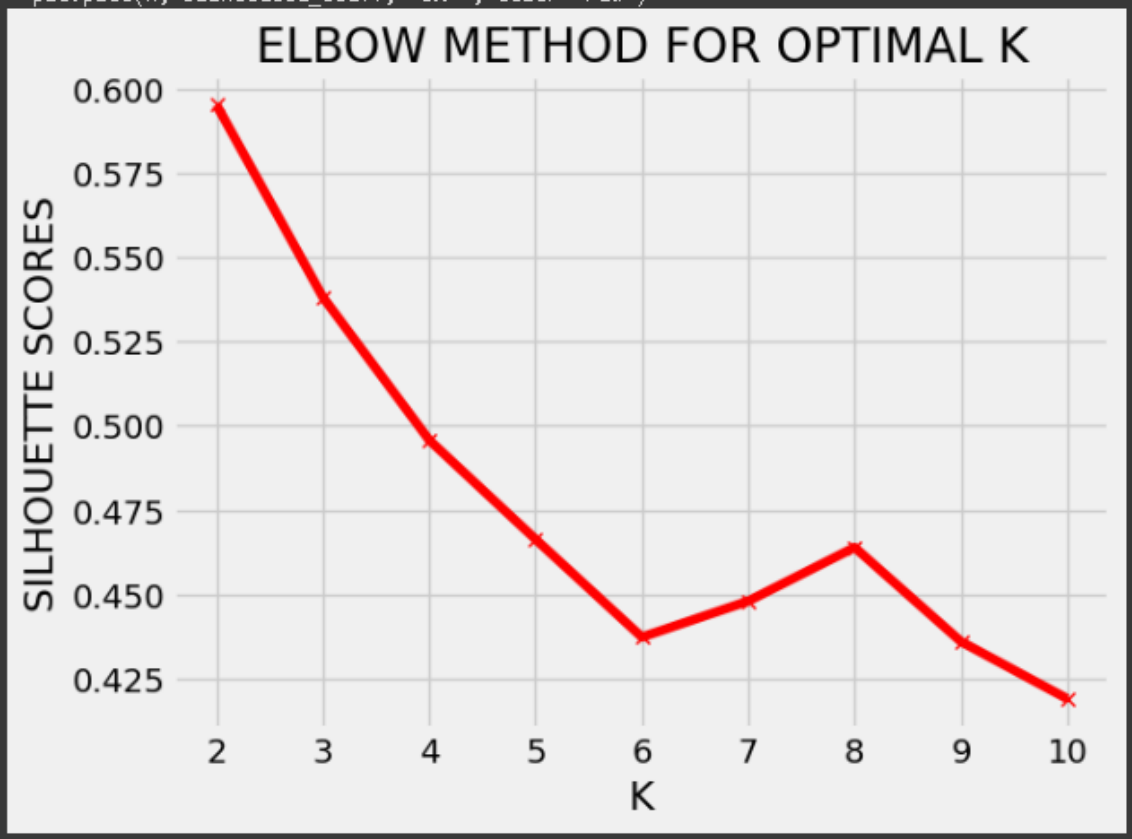


```
[ ] # GETTING HIGHEST SSE BY SILHOUETTE SCORE AND PLOTTING GRAPH
    from sklearn.metrics import silhouette_score
```

```
[ ] silhouette_coeff = []
    K = range(2,11)
    for k in range(2, 11):
        kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
        kmeans.fit(train_features)
        score = silhouette_score(train_features, kmeans.labels_)
        silhouette_coeff.append(score)
```

```
[ ] #VISUALIZING PLOT
    plt.style.use("fivethirtyeight")
    plt.plot(K, silhouette_coeff, 'bx-', color='red')
    plt.xlabel('K')
    plt.ylabel('SILHOUETTE SCORES')
    plt.title('ELBOW METHOD FOR OPTIMAL K')
    plt.show()
```

```
<ipython-input-182-5f3048cd3aa6>:3: UserWarning: color is redundantly defined by the 'color' key
  plt.plot(K, silhouette_coeff, 'bx-', color='red')
```



```
[ ] # FINDING PREDICTED LABELS
    y_pred = kmeans.predict(train_features)
    print(y_pred)
```

```
[4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 4 4 4 4 4 4 4 9 9 4 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 5 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 5 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 5
 8 8 8 8 8 8 8 8 8 5 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 5 8 5 8 8 8 8 8 5 5 5 8 8 8 5 6 6 6 6 6 6 5 5 6 6 6 6 6 5 6 6 6 6 6 6 5
 6 5 6 6 5 6 6 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6 6 6 5 6 6 6 5 6 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 5 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 3 7 7 7 7 7 7 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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 3 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 3 0 0
 0 0 0 0 0 3 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 3 0 0
 0 0 3 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 2 2 2 2 2 2
 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2]
```

```
# CREATING 3 CLUSTERS WITH INIT RANDOM
kmeans = KMeans(init="k-means++",n_clusters=3,n_init=10,max_iter=300,random_state=42)
```

```
# FITTING THE SCALED FEATURES
kmeans.fit(test_features)
```

```
KMeans
KMeans(n_clusters=3, n_init=10, random_state=42)
```

```
# GETTING POSTION OF CENTRIODS OF CLUSTERS
kmeans.cluster_centers_
```

```
array([[9.63000000e+02, 2.30769231e+00, 6.15384615e-01, 3.02167832e+01,
        4.47552448e-01, 3.28671329e-01, 3.68282650e+01],
       [1.24200000e+03, 2.22962963e+00, 6.59259259e-01, 2.97259259e+01,
        4.59259259e-01, 4.51851852e-01, 3.79744452e+01],
       [1.10450000e+03, 2.25714286e+00, 6.35714286e-01, 2.88464286e+01,
        4.35714286e-01, 4.00000000e-01, 3.20753257e+01]])
```

```
# FINDING LOWEST SSE
print("LOWEST SSE OF ALL 3 CLUSTERS IS:", kmeans.inertia_)
```

```
LOWEST SSE OF ALL 3 CLUSTERS IS: 2043267.877350005
```

```
# NUMBER OF ITERATIONS
kmeans.n_iter_
```

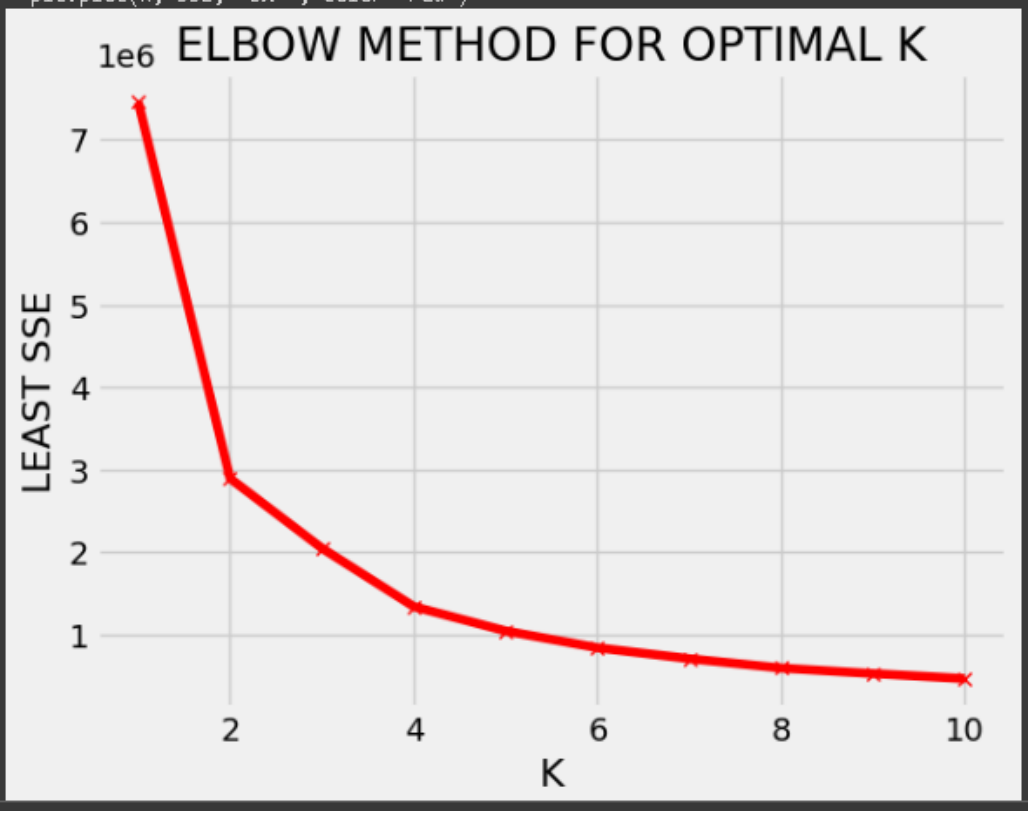
```
13
```

```
SSE = []
K = range(1,11)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(test_features)
    SSE.append(km.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

```
#VISUALIZING PLOT
plt.style.use("fivethirtyeight")
plt.plot(K, SSE, 'bx-', color='red')
plt.xlabel('K')
plt.ylabel('LEAST SSE')
plt.title('ELBOW METHOD FOR OPTIMAL K')
plt.show()
```

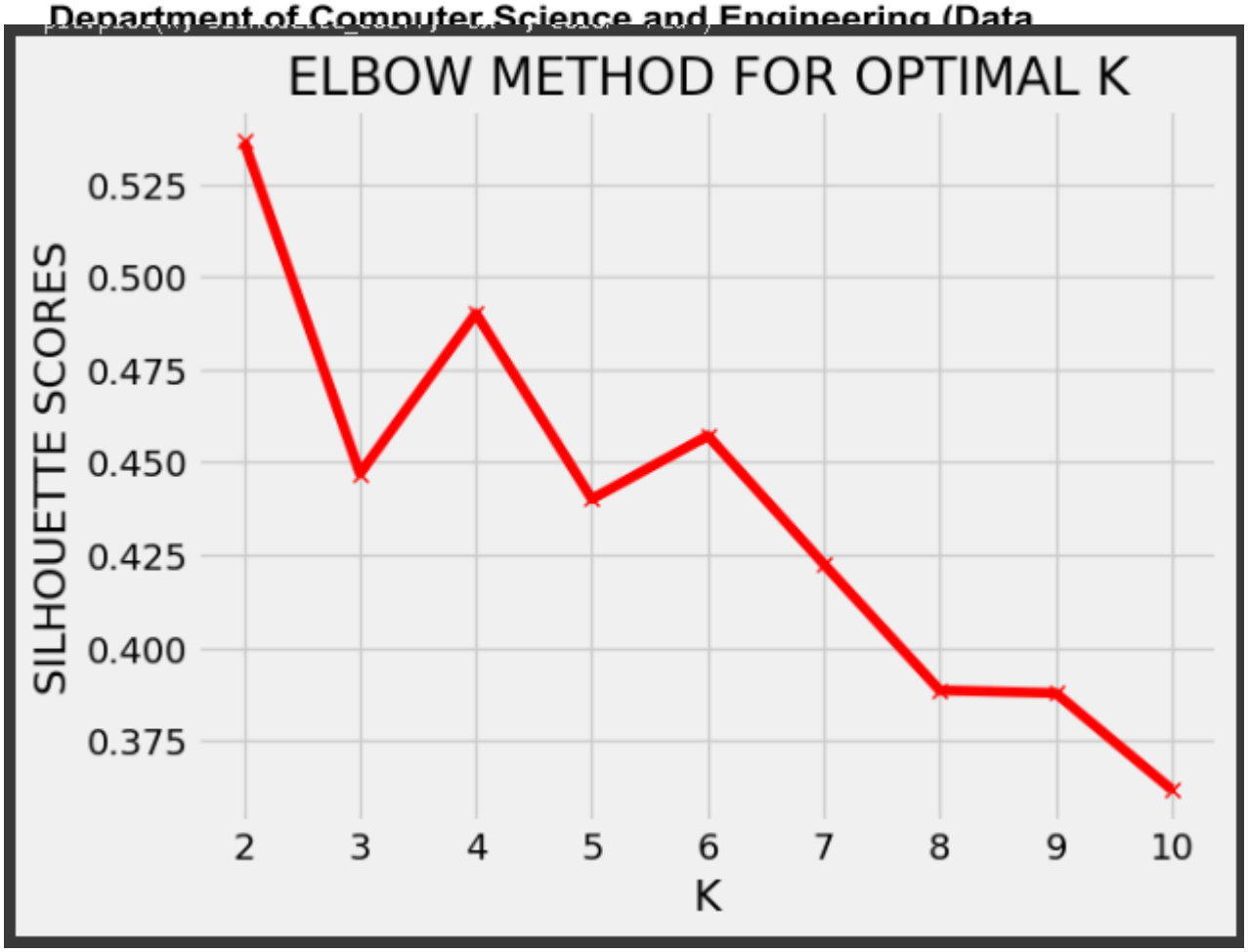
```
<ipython-input-190-26834c19a141>:3: UserWarning: color is redundantly defined by the 'color' keywo
  plt.plot(K, SSE, 'bx-', color='red')
```



```
[ ] # GETTING HIGHEST SSE BY SILHOUETTE SCORE AND PLOTTING GRAPH
    from sklearn.metrics import silhouette_score
```

```
[ ] silhouette_coeff = []
    K = range(2,11)
    for k in range(2, 11):
        kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
        kmeans.fit(test_features)
        score = silhouette_score(test_features, kmeans.labels_)
        silhouette_coeff.append(score)
```

```
[ ] #VISUALIZING PLOT
    plt.style.use("fivethirtyeight")
    plt.plot(K, silhouette_coeff, 'bx-', color='red')
    plt.xlabel('K')
    plt.ylabel('SILHOUETTE SCORES')
    plt.title('ELBOW METHOD FOR OPTIMAL K')
    plt.show()
```




```
[ ] # FINDING PREDICTED LABELS
    y_pred = kmeans.predict(test_features)
    print(y_pred)
```

```
[3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 1 5 5 5 5 5 1 5 5 5 5 1 5 5 5 5 1 5 5 5 5
 1 1 5 5 5 5 5 1 5 5 5 5 5 5 5 5 5 5 5 5 5 4 5 5 5 4 5 5 5 5 4 4 4 4 4 4 4
 4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 4 4 4 4 4
 4 4 4 4 4 4 4 4 1 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
 2 2 2 2 2 2 2 2 2 2 2 9 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 9 1 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 9 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 9 0 9 0 0 0 0 0 0 0 0 0 0 0 7 7 9 7 7 7 7 7 9 7 7
 7 7 7 7 7 7 7 7 7 9 7 9 7 7 7 7 7 9 7 9 7 7 7 7 7 7 9 7 7 9 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 8 6 7 7 7 7 7 7 8 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 9 8 8 8 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 8 8 8 8 8 8
 6 8 8 8 8 8 8 8 8 8 8]
```