CSE-3024 Web Mining

Lab Assignment 9

Alokam Nikhitha

19BCE2555

K-Means

Problem statement:

Illustrate the k-means clustering to cluster the data points for at least five epoch properly.

How to Implementing K-Means Clustering?

- Using the elbow method to determine the optimal number of clusters for kmeans clustering
- Visualising the clusters
- Plotting the centroids of the clusters

Dataset used:

- Shopping-data
- https://archive.ics.uci.edu/ml/machine-learning-databases/

Procedure:

- Import necessary libraries - sklearn, numpy, pandas, etc.

-Using pandas, we first import the dataset into our workspace.

- Select the number of clusters for the dataset (K)

- Select K number of centroids

- By calculating the Euclidean distance or Manhattan distance assign the points to the nearest centroid, thus creating K groups

- Now find the original centroid in each group

- Again reassign the whole data point based on this new centroid, then repeat step 4 until the position of the centroid doesn’t change.

- Using the elbow method to determine the optimal number of clusters for kmeans clustering

- Visualising the clusters and Plotting the centroids of the clusters

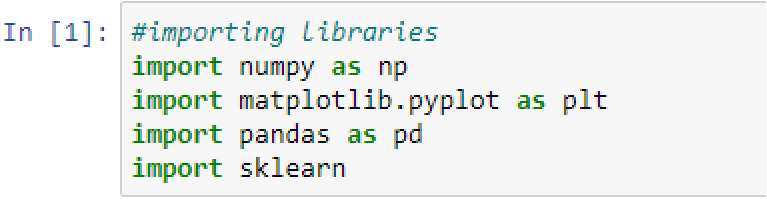
Code:

```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Importing the Datasets
dataset = pd.read_csv('shopping-data.csv')
X = dataset.iloc[:, 3:].values


#Elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
   kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300,
n_init=10)
```

```
    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()


#Applying Kmeans to the dataset
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300,
n_init=10);
y_kmeans = kmeans.fit_predict(X)


#Printing out the cluster each input belongs to
y_kmeans


# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c =
'red', label = 'Standard Customers')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c =
'blue', label = 'Careless Customers')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c =
'cyan', label = 'Target Customers')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c =
'magenta', label = 'Sensible Customers')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c =
'green', label = 'Careful Customers')
plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label =
'Centroids')
plt.title ('Clusters of Clients')
plt.xlabel ('Annual Income (k$)')
plt.ylabel ('Spending Score (1-100)')
plt.legend()
```

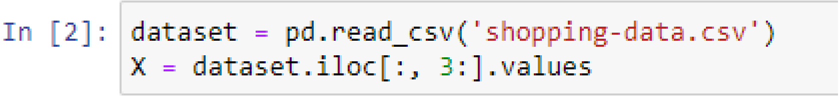
plt.show()

Code Snippets and Outputs:



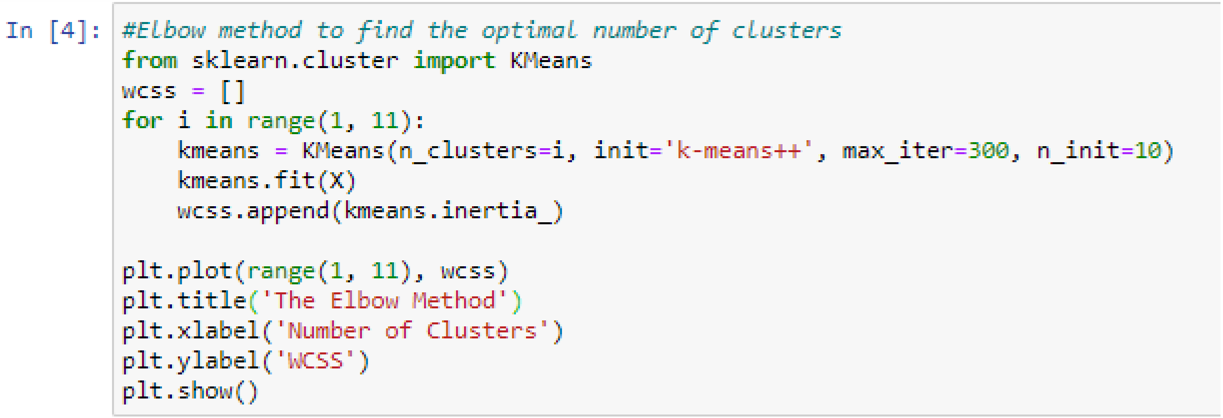
```
In [1]: #importing libraries
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import sklearn
```

Import necessary libraries - sklearn, numpy, pandas, etc.

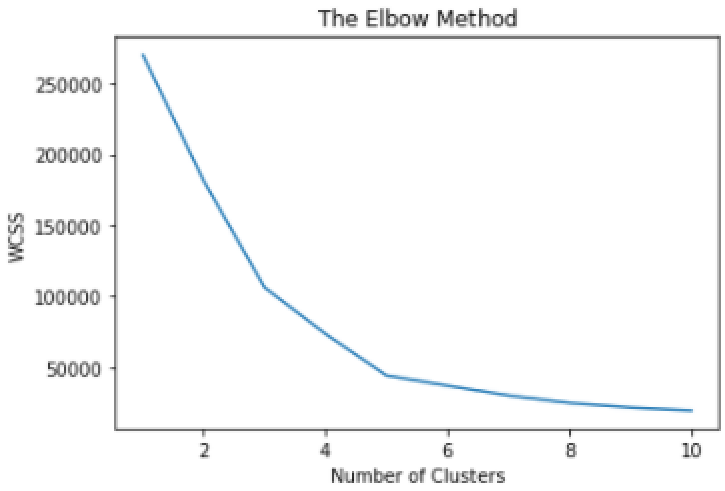


```
In [2]: dataset = pd.read_csv('shopping-data.csv')
        X = dataset.iloc[:, 3:].values
```

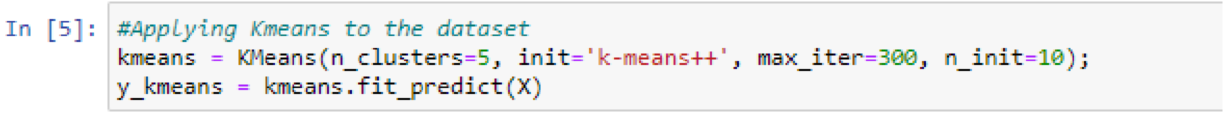
Using pandas, we first import the dataset into our workspace and are assigning the income attribute along with shopping score as independent variables.



```
In [4]: #Elbow method to find the optimal number of clusters
        from sklearn.cluster import KMeans
        wcss = []
        for i in range(1, 11):
            kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10)
            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()
```

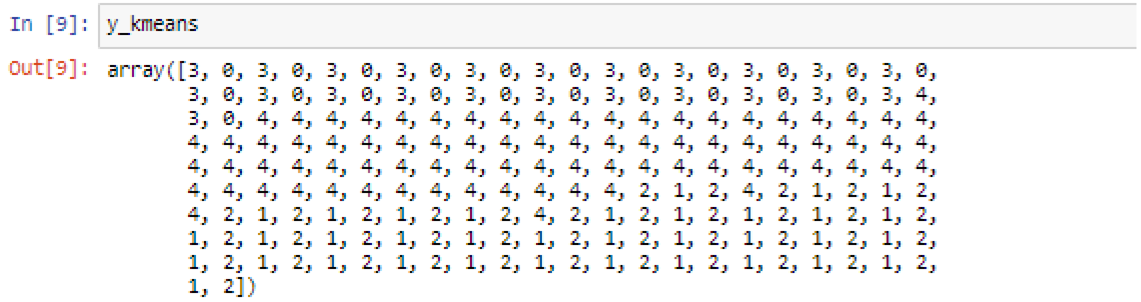


Here we are plotting a graph that marks Within Cluster Sum of Squares (WCSS) with the increase in number of clusters. We can see an elbow formation when the number of clusters is 5 and hence, we assume that optimal number of clusters in our dataset is 5



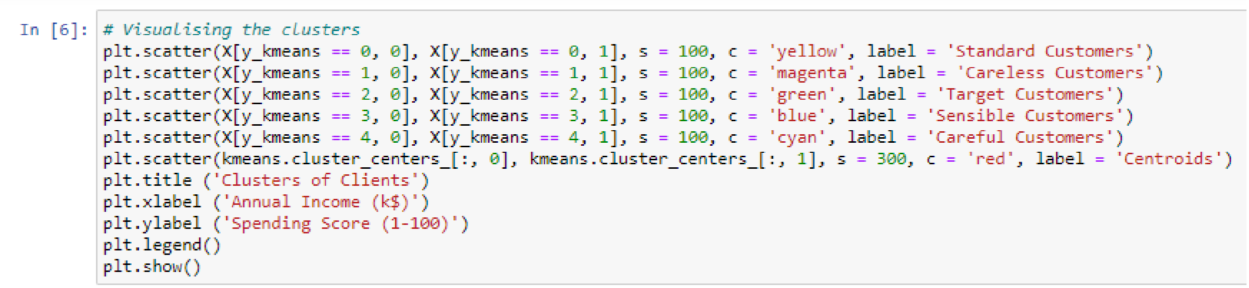
```
In [5]: #Applying Kmeans to the dataset
        kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300, n_init=10);
        y_kmeans = kmeans.fit_predict(X)
```

Here we are training our k-means model with 5 clusters. We are also generating the y_kmeans array that stores the cluster index of each input attribute from 0 to 4

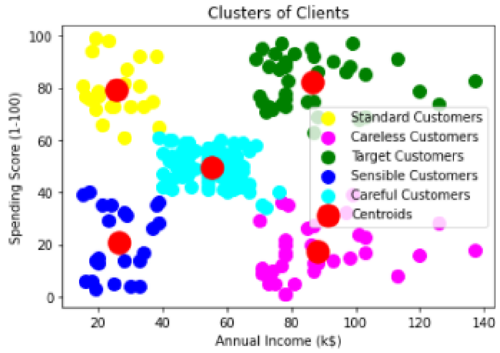


```
In [9]: y_kmeans

Out[9]: array([3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0,
               3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 4,
               3, 0, 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, 2, 1, 2, 4, 2, 1, 2, 1, 2,
               4, 2, 1, 2, 1, 2, 1, 2, 1, 2, 4, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
               1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
               1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
               1, 2])
```

Here we are printing our y_kmeans array and we can see that each input cell is assigned a value between 0 and 4, both inclusive. This corresponds to the cluster index of each input.



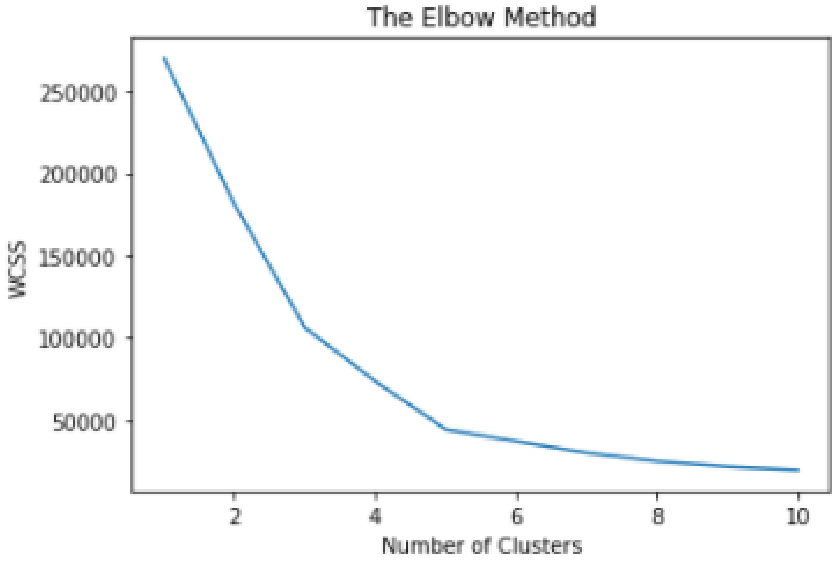
```
In [6]: # Visualising the clusters
        plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'yellow', label = 'Standard Customers')
        plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'magenta', label = 'Careless Customers')
        plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Target Customers')
        plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'blue', label = 'Sensible Customers')
        plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'cyan', label = 'Careful Customers')
        plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'red', label = 'Centroids')
        plt.title ('Clusters of Clients')
        plt.xlabel ('Annual Income (k$)')
        plt.ylabel ('Spending Score (1-100)')
        plt.legend()
        plt.show()
```



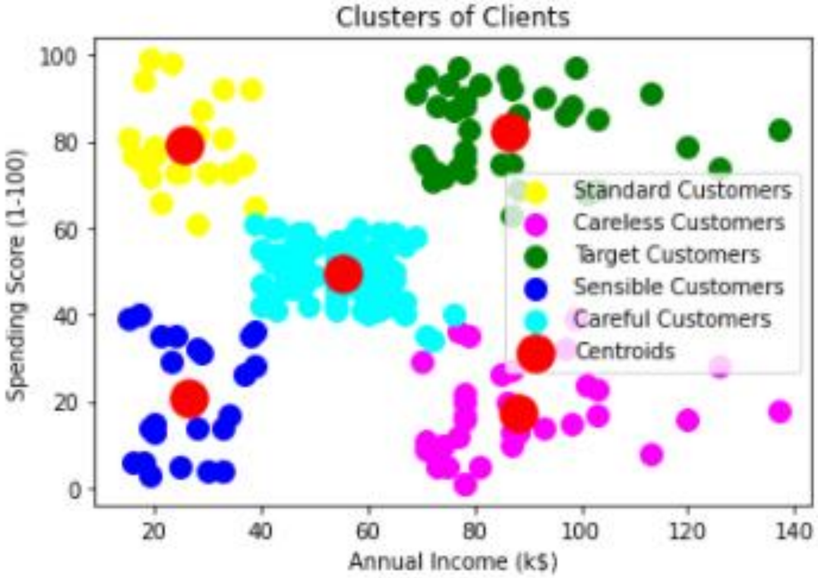
Here we have visualized our results. We have labelled different clusters as blue, green, pink, yellow and cyan. Each cluster correspond to different category of target audience. We have also marked centroids of each cluster which are red in colour.

Results and Output

Elbow Method Graph



Clustering Graph:



Here different clusters are marked as blue, yellow, magenta, cyan and green. The red dot over each cluster represents its centroid.

We can categorise these clusters as: -

- Yellow Cluster corresponds to careless customers as they have low income but high spending.
- Blue Cluster as Sensible customers, becoz they have low income and low spending.
- Cyan Clusters are standard cluster that suggest they have median income and median spending.
- The pink coloured cluster correspond to Target Customers, as they have high income but low spending, the shopping company can give them offers and attractions as they are capable of spending more but they aren't doing it currently.
- Finally, the Green coloured clusters are Careful customers. They have high income and thus high spending as well.