Major Project Synopsis

On

Customer Segmentation using Machine Learning

In partial fulfilment of requirements for the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by :

Prachi Gothwal [20100BTCSDSI07284]

Sakshi Patel [20100BTCSDSI07291]

Under the guidance of

Prof. Om Kant Sharma

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALA, INDORE

JUL-DEC – 2022

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

INTRODUCTION

In this Data Science Project, we will perform one of the most essential applications of machine learning – Customer Segmentation. In this project, we will implement customer segmentation . Whenever we need to find your best customer, customer segmentation is the ideal methodology. Also, in this data science project, we will see the descriptive analysis of our data and then implement several versions of the K-means algorithm. So, follow the complete data science customer segmentation project using machine learning in R and become a pro in Data Science.

PROBLEM STATEMENT

Consider a mall a very famous mall and we are a very experienced data scientist and this mall wants all information about their customers and other expects. As a data scientist you can build a system that can cluster customer in different groups.

One group of customer are those who purchase more from that mall and other group are those who don't purchase too much from that mall. So having these group of customer these is easy to understand and better details to make and understand marketing strategies.

WORK FLOW

First we want to make a small customer data to train or instruct our machine learning project model. So first is getting those customer data and then we have to process these data, we cannot feed these data directly in machine learning model so we need to select key features that particular dataset contain is all come in analysis process and after that we need to choose the current no.

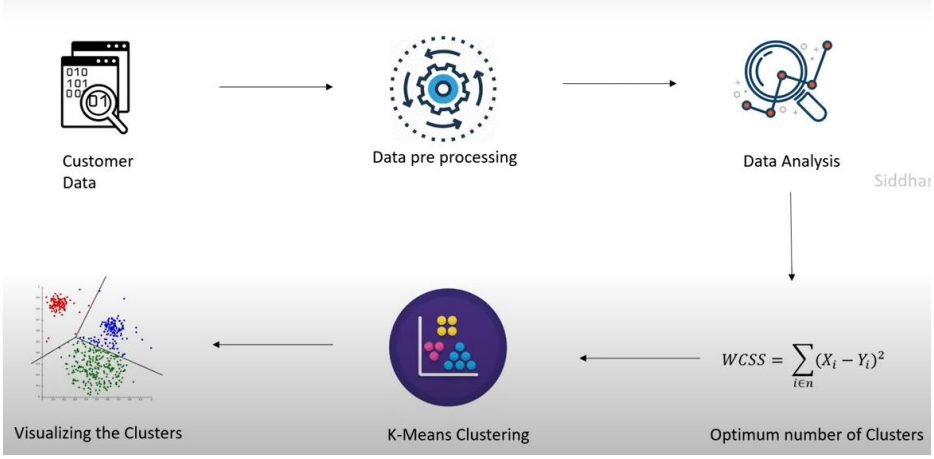
Of clusters and tell the machine learning project how many clusters are exists.

So we find the no. of clusters using method within cluster sum of square, so in this process we find the value of WCSS.

Once we have no. of cluster we can find the data in K-mean cluster Algorithm.

So once we f feed this algorithm to this model it can grew the data dependent on the similarities or similar expending pattern etc.

After that we can visualizing these cluster data by putting these data on the prediction made by the clustering models to get better insect of data.



K-Means Algorithm

- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters.
- The goal of K means is to group data points into distinct non-overlapping subgroups.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.
- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

Step-1: Data pre-processing Step

Importing the Dependencies

In the above code, the numpy we have imported for the performing mathematics calculation, **matplotlib** is for plotting the graph, and **pandas** are for managing the dataset.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.cluster import KMeans
```

Importing dataset.

Next, we will import the dataset that we need to use. So here, we are using the Mall_Customer_data.csv dataset. It can be imported using the below code:

```
In [7]: #read dataset
        print("Prachi Gothwal")
        print("Sakshi Patel")
        customer_data = pd.read_csv("C:\\Users\\Abhishek\\Desktop\\Mall_Customers.csv")

        Prachi Gothwal
        Sakshi Patel
```

Data Analysis

```
In [8]: print("Prachi Gothwal")
        print("Sakshi Patel")
        customer_data.head()

        Prachi Gothwal
        Sakshi Patel
```

Out[8]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [9]:
```
# finding the number of rows and columns
print("Prachi Gothwal")
print("Sakshi Patel")
customer_data.shape
```

```
Prachi Gothwal
Sakshi Patel
```

Out[9]: (200, 5)

---

[10]:
```
print("Prachi Gothwal")
print("Sakshi Patel")
customer_data.info()
```

```
Prachi Gothwal
Sakshi Patel
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

[12]:
```
# checking for missing values
print("Prachi Gothwal")
print("Sakshi Patel")
customer_data.isnull().sum()
```

```
Prachi Gothwal
Sakshi Patel
```

Out[12]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Choosing the Annual Income Column & Spending Score column

```
In [13]: print("Prachi Gothwal")
         print("Sakshi Patel")
         X = customer_data.iloc[:,[3,4]].values

         Prachi Gothwal
         Sakshi Patel
```

```
In [14]: print(X)

         [[ 15  39]
          [ 15  81]
          [ 16   6]
          [ 16  77]
          [ 17  40]
          [ 17  76]
          [ 18   6]
          [ 18  94]
          [ 19   3]
          [ 19  72]
          [ 19  14]
          [ 19  99]
          [ 20  15]
          [ 20  77]
          [ 20  13]
          [ 20  79]
          [ 21  35]
          [ 21  66]
          [ 23  29]
```

Step-2: Finding the optimal number of clusters using the elbow method

In the second step, we will try to find the optimal number of clusters for our clustering problem.

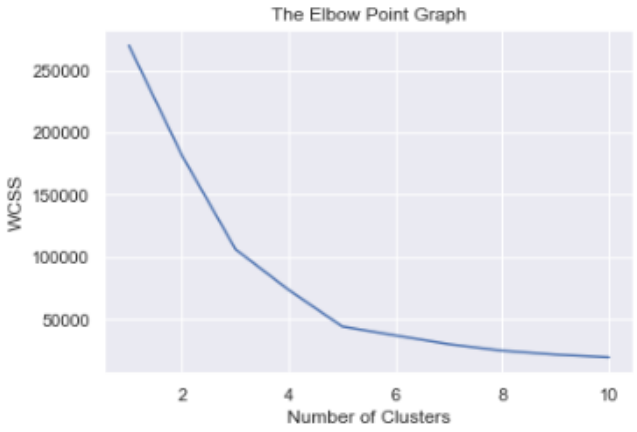
- Choosing the number of clusters
- WCSS -> Within Clusters Sum of Squares

In [15]:
```
# finding wcss value for different number of clusters
print("Prachi Gothwal")
print("Sakshi Patel")
wcss = []

for i in range(1,11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
  kmeans.fit(X)

  wcss.append(kmeans.inertia_)
```

```
Prachi Gothwal
Sakshi Patel
C:\Users\Abhishek\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory le
ak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```

In [16]:
```
# plot an elbow graph
print("Prachi Gothwal")
print("Sakshi Patel")
sns.set()
plt.plot(range(1,11), wcss)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

```
Prachi Gothwal
Sakshi Patel
```



- From the above plot, we can see the elbow point is at 5. So the number of clusters here will be 5.
- Optimum Number of Clusters = 5

Step- 3: Training the K-means algorithm on the training dataset

As we have got the number of clusters, so we can now train the model on the dataset.

To train the model, we will use the same two lines of code as we have used in the above section, but here instead of using i, we will use 5, as we know there are 5 clusters that need to be formed. The code is given below:

```
In [17]: print("Prachi Gothwal")
         print("Sakshi Patel")
         kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0)
         # return a label for each data point based on their cluster
         Y = kmeans.fit_predict(X)
         print(Y)

         Prachi Gothwal
         Sakshi Patel
         [4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
          3 4 3 4 3 4 1 4 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 2 0 2 1 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 1 2 0 2 0 2
          0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
          2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]
```

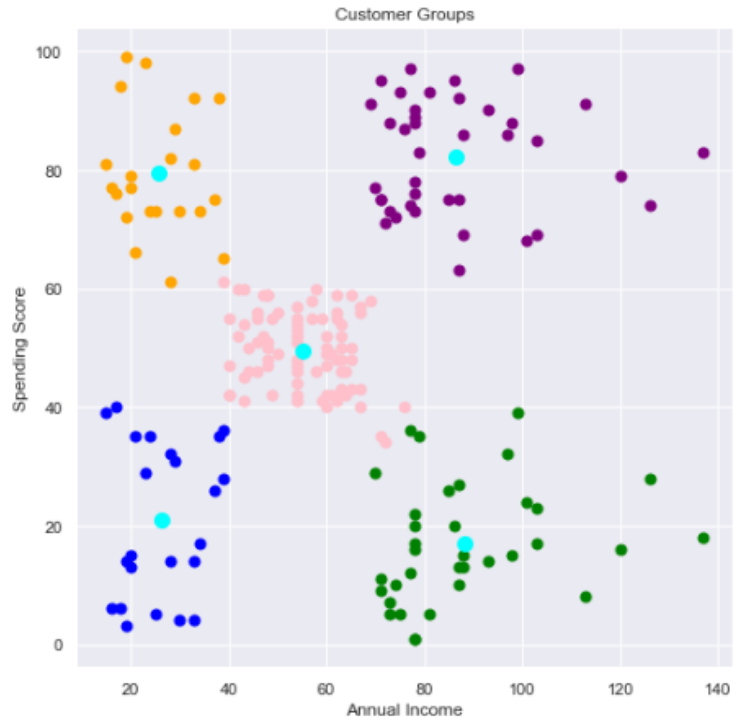
5 Clusters - 0, 1, 2, 3, 4

Step-4: Visualizing the Clusters

The last step is to visualize the clusters. As we have 5 clusters for our model, so we will visualize each cluster one by one.

```
# plotting all the clusters and their Centroids
print("Prachi Gothwal")
print("Sakshi Patel")
plt.figure(figsize=(8,8))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='pink', label='Cluster 2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='purple', label='Cluster 3')
plt.scatter(X[Y==3,0], X[Y==3,1], s=50, c='orange', label='Cluster 4')
plt.scatter(X[Y==4,0], X[Y==4,1], s=50, c='blue', label='Cluster 5')

# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='cyan', label='Centroids')
plt.title('Customer Groups')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```

Prachi Gothwal
Sakshi Patel



As we see we have multiple cluster here in different different color. All the cluster are portioned differently on the screen and the cluster represent the group of customer we are getting .As we focus on particular cluster it means a group of people we can see not too many have more annual income but they are regular customer of the mall except these others are not regular customers of mall.

HIERARCHICAL CLUSTERING

Now the same task will be implemented using Hierarchical clustering. The reading of CSV files and creating a dataset for algorithms will be common as given in the first and second step. In K-Means, the number of optimal clusters was found using the elbow method.

- Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as hierarchical cluster analysis or HCA.
- In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.
- Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

Step-1: Data pre-processing Step

Step 1.1. Importing the Dependencies

In the above code, the numpy we have imported for the performing mathematics calculation, **matplotlib** is for plotting the graph, and **pandas** are for managing the dataset.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.cluster import KMeans
```

Step 1.2. Importing dataset.

Next, we will import the dataset that we need to use. So here, we are using the Mall_Customer_data.csv dataset.

It can be imported using the below code:

```
In [7]: #read dataset
        print("Prachi Gothwal")
        print("Sakshi Patel")
        customer_data = pd.read_csv("C:\\Users\\Abhishek\\Desktop\\Mall_Customers.csv")

        Prachi Gothwal
        Sakshi Patel
```

After the importing of dataset we now analysis the data as we have done above for the K-Means clustering algorithm.

Step 1.3. Data Analysis

```
In [8]: print("Prachi Gothwal")
        print("Sakshi Patel")
        customer_data.head()
```

```
Prachi Gothwal
Sakshi Patel
```

Out[8]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
In [9]: # finding the number of rows and columns
        print("Prachi Gothwal")
        print("Sakshi Patel")
        customer_data.shape
```

```
Prachi Gothwal
Sakshi Patel
```

Out[9]: (200, 5)

```
[10]: print("Prachi Gothwal")
      print("Sakshi Patel")
      customer_data.info()
```

```
Prachi Gothwal
Sakshi Patel
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
In [12]: # checking for missing values
         print("Prachi Gothwal")
         print("Sakshi Patel")
         customer_data.isnull().sum()

         Prachi Gothwal
         Sakshi Patel

Out[12]: CustomerID                0
         Gender                    0
         Age                       0
         Annual Income (k$)        0
         Spending Score (1-100)    0
         dtype: int64
```

Choosing the Annual Income Column & Spending Score column

```
In [13]: print("Prachi Gothwal")
         print("Sakshi Patel")
         X = customer_data.iloc[:,[3,4]].values

         Prachi Gothwal
         Sakshi Patel

In [14]: print(X)

         [[ 15  39]
          [ 15  81]
          [ 16   6]
          [ 16  77]
          [ 17  40]
          [ 17  76]
          [ 18   6]
          [ 18  94]
          [ 19   3]
          [ 19  72]
          [ 19  14]
          [ 19  99]
          [ 20  15]
          [ 20  77]
          [ 20  13]
          [ 20  79]
          [ 21  35]
          [ 21  66]
          [ 23  29]
```

Step-2: Finding the optimal number of clusters using the elbow method

In the second step, we will try to find the optimal number of clusters for our clustering problem.

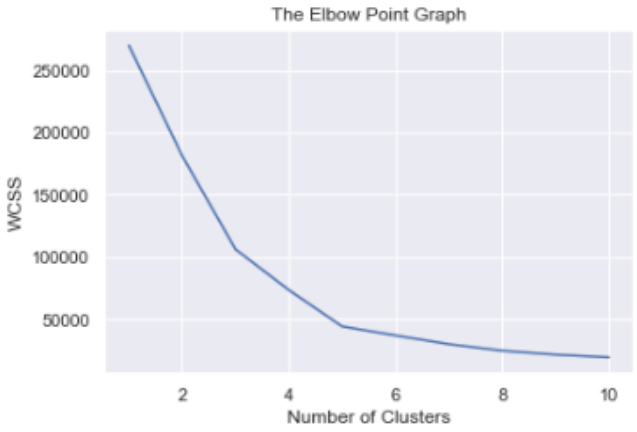
- Choosing the number of clusters
- WCSS -> Within Clusters Sum of Squares

In [15]:
```
# finding wcss value for different number of clusters
print("Prachi Gothwal")
print("Sakshi Patel")
wcss = []

for i in range(1,11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
  kmeans.fit(X)

  wcss.append(kmeans.inertia_)
```

```
Prachi Gothwal
Sakshi Patel
C:\Users\Abhishek\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory le
ak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```

In [16]:
```
# plot an elbow graph
print("Prachi Gothwal")
print("Sakshi Patel")
sns.set()
plt.plot(range(1,11), wcss)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

```
Prachi Gothwal
Sakshi Patel
```

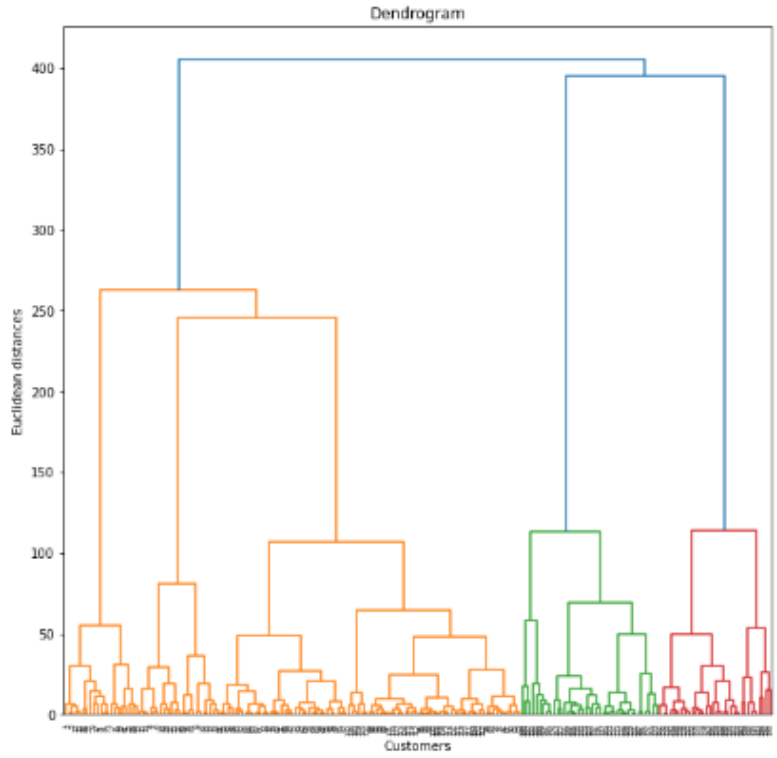


- From the above plot, we can see the elbow point is at 5. So the number of clusters here will be 5.
- Optimum Number of Clusters = 5

In hierarchical clustering, the dendrograms are used for this purpose. The below lines of code plot a dendrogram for our dataset.

```
In [2]: print("Prachi Gothwal")
        print("Sakshi Patel")
        import scipy.cluster.hierarchy as sch
        import matplotlib.pyplot as plt
        import pandas as pd
        import numpy as np
        import seaborn as sns
        customer_data = pd.read_csv("C:\\Users\\Abhishek\\Desktop\\Mall_Customers.csv")
        plt.figure(figsize=(10,10))
        X = customer_data.iloc[:,[3,4]].values
        dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
        plt.title('Dendrogram')
        plt.xlabel('Customers')
        plt.ylabel('Euclidean distances')
        plt.show()
```

```
Prachi Gothwal
Sakshi Patel
```



If you are aware of this method, you can see in the above diagrams. The combination of 5 lines are not joined on the Y-axis from 100 to 240, for about 140 units..

So, the optimal number of clusters will be 5 for hierarchical clustering.

Step- 3: Training the K-means algorithm on the training dataset

As we have got the number of clusters, so we can now train the model on the dataset.

Now we train the hierarchical clustering algorithm and predict the cluster for each data point.

```
In [5]: print("Prachi Gothwal")
        print("Sakshi Patel")
        from sklearn.cluster import AgglomerativeClustering
        hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
        y_hc = hc.fit_predict(X)
```

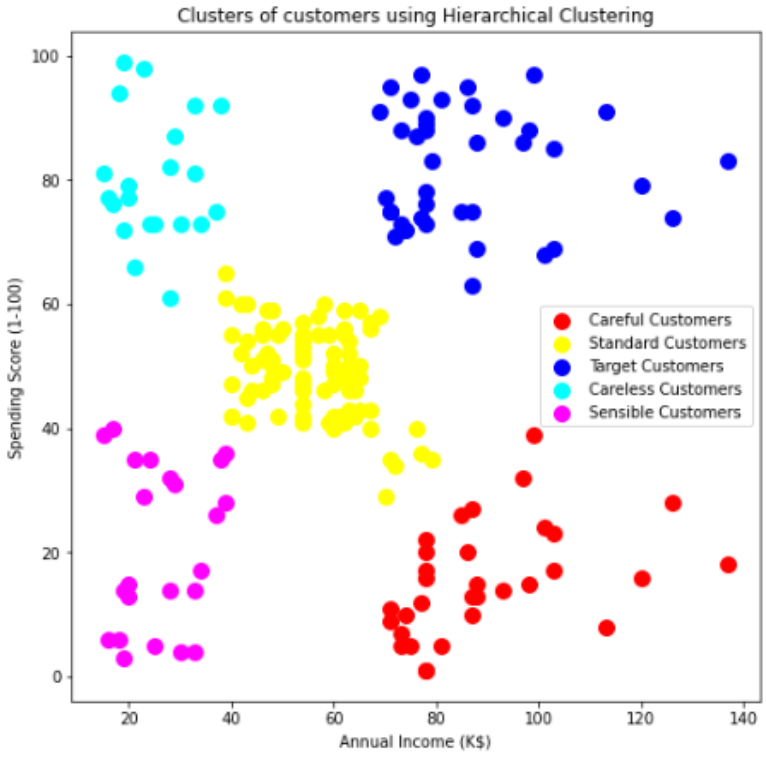
```
Prachi Gothwal
Sakshi Patel
```

Step-4: Visualizing the Clusters

Once the algorithm predicts the cluster for each of the data points, it can be visualized now.

```
In [6]: print("Prachi Gothwal")
        print("Sakshi Patel")
        plt.figure(figsize=(8,8))
        plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Careful Customers')
        plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'yellow', label = 'Standard Customers')
        plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'blue', label = 'Target Customers')
        plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Careless Customers')
        plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Sensible Customers')
        plt.title('Clusters of customers using Hierarchical Clustering')
        plt.xlabel('Annual Income (K$)')
        plt.ylabel('Spending Score (1-100)')
        plt.legend()
        plt.show()
```

```
Prachi Gothwal
Sakshi Patel
```



- We can see in the above diagram that clustering of customers is almost similar to what was done by K-Means clustering. Only the colour combinations have been changed for differentiating both the diagrams.

CONCLUSION

So through these survey we can focus on that group of customer who are not buying to much from the mall, we can give them offers so they can be regular customers.

Through these survey we can improve the market value and the profit of mall and improve their status.

Comparison Between K-Means & Hierarchical Clustering.

- As we have seen above , the results of both the clustering are almost similar to the same dataset. It may be possible that when we have a very large dataset, the shape of clusters may differ a little.

- However, along with many similarities, these two techniques have some differences also.

- The below table shows the comparison between K-Means and Hierarchical clustering algorithms based on our implementations.

	K-Means Clustering	**Hierarchical Clustering**
Category	Centroid based, partition-based	Hierarchical, Agglomerative
Method to find the optimal number of clusters	The Elbow method using WCSS	Dendrogram
Directional approach	Not any, the only centroid is considered to form clusters	Top-down, bottom-up
Python Library	sklearn – KMeans	sklearn-AgglomerativeClustering