HO CHI MINH UNIVERSITY OF SCIENCE

Faculty of Information Technology

Project 3

Subject: Data Mining

Topic: CLUSTERING

Instructors:

- Thầy Lê Ngọc Thành
- Thầy Nguyễn Thái Vũ

1. Group information, Rate of division of work, level of completion

Student ID	Full name	Tasks	Level of complete
19127399	Huỳnh Cao Nhật Hiếu	1. Data preprocessing, Discovering. 2. Implement KMean algorithm. 3. Write report.	100%
19127444	Ngô Đăng Khoa	1. Implement DBScan. 2. Write report.	100%
19127134	Nguyễn Gia Hân	1. Data discovering. 2. Implement Hierarchy algorithm. 3. Write report.	100%

2. Brief introduction

- Customer Segmentation is the subdivision of a market into discrete customer groups that share similar characteristics. Customer Segmentation can be a powerful means to identify unsatisfied customer needs. Using the above data companies can then outperform the competition by developing uniquely appealing products and services.
- In this project, my team will preprocess, discover data and use 3 clustering algorithms to cluster this dataset.

3. Data processing

```
In [5]: df.isnull().sum()
```

```
Out[5]: CustomerID              0
        Gender                  0
        Age                     0
        Annual Income (k$)      0
        Spending Score (1-100)  0
        dtype: int64
```

- There is no missing values in the dataset.

I dropped the `CustomerID` column as that does not seem relevant to the context.

```
In [6]: df.drop(['CustomerID'], axis=1,inplace=True)
        df
```

Out[6]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40
...	...	...	...	...
195	Female	35	120	79
196	Female	45	126	28
197	Male	32	126	74
198	Male	32	137	18
199	Male	30	137	83

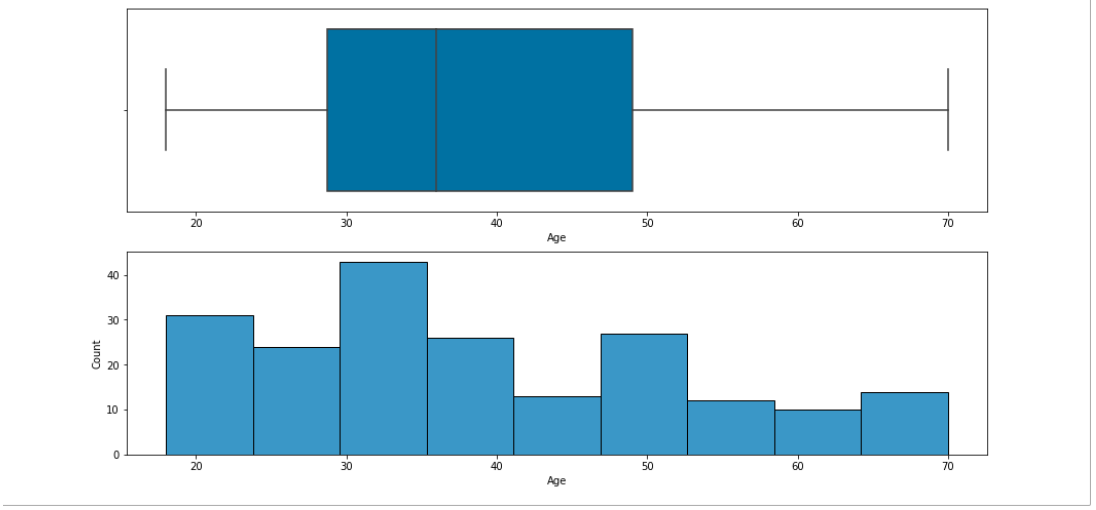
200 rows × 4 columns

- The dataset has no null value so we do not need to handle missing values.
- We find that the 'CustomerID' column does not seem relevant to the context and has no value to the algorithms we implement, therefore, we decide to drop this column.

4. Discovering

a. Attribute 'Age'

```
In [7]: figure1, axis1 = plt.subplots(2, 1,figsize=(15,8))
        sns.boxplot(x='Age',data=df,ax=axis1[0])
        sns.histplot(x='Age',data=df,ax=axis1[1])
        plt.show()
```

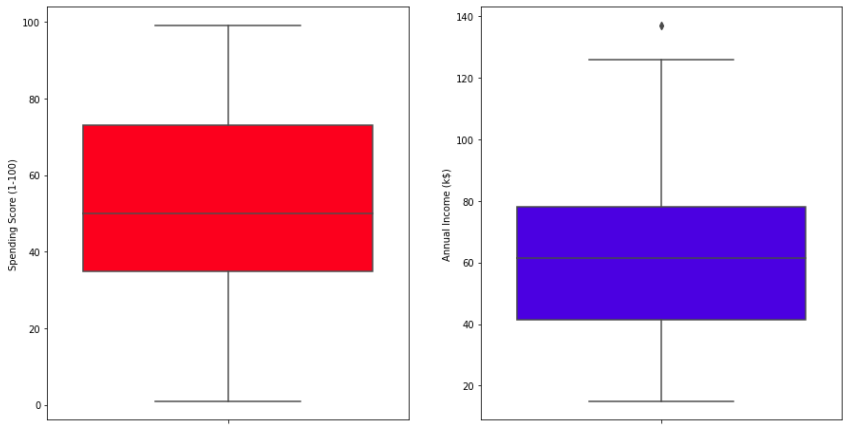


- As we can see in 2 graph above, there are some information about Age column:

- The average age of the customers in the data is about 39.
- Age in the range from 30 to 50 accounts for a large quantity in the data.

b. Attribute Spending Score (1-100) and Annual Income (k$)

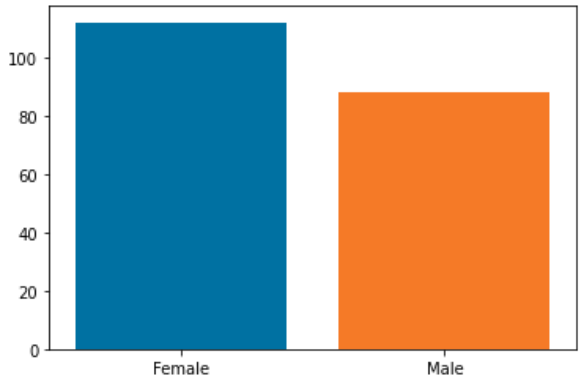
```
In [8]: figure2, axis2 = plt.subplots(1, 2, figsize=(15,8))

sns.boxplot(y="Spending Score (1-100)", data=df, color='red', ax=axis2[0])
sns.boxplot(y="Annual Income (k$)", data=df, color='blue', ax=axis2[1])

plt.show()
```



- According to a graph illustrating Spending Score, the value is distributed in the range of 35 - 75 (scores) and the median of Spending Score gets the value of 50.
- On the other hand, with graph illustrating Annual Income(k$), the value is distributed from 40 to 80(k$) and the median gets 60(k$)

c. Attribute Gender

```
sns.barplot(x=df.Gender.value_counts().index, y=df.Gender.value_counts().values)

plt.show()
```



- Bar chart that illustrates the quantity of Male and Female shows us that females in this dataset take more surveys than Male customers.

d. Attribute Annual Income vs Age and Spending Score

```
In [40]: fig, ax = plt.subplots(figsize=(15,8))

sns.lineplot(x='Annual Income (k$)',y='Age',data=df, color = 'green', label='Age',ax=ax)
sns.lineplot(x='Annual Income (k$)',y='Spending Score (1-100)',data=df, color = 'orange',label='Spending Score',ax=a
ax.set_title('Annual Income vs Age and Spending Score', fontsize = 15)

ax.set_ylabel('')

ax.legend()
plt.show()
```

- The above plot between Annual Income and Age represented by a green color line, and a plot between Annual Income and the Spending Score represented by an orange color, shows how Age and Spending Varies with Annual Income.
- Through the graph, we find the line between Annual Income & Age as well as Annual Income & Spending Score fluctuate continuously, therefore it is quite difficult to find the insight from this graph.

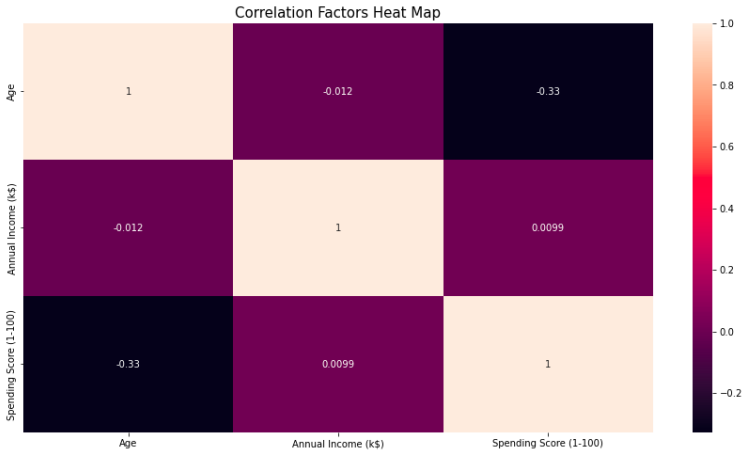
e. Attribute Gender vs Spending Score

- There are more males who get paid than females. But, The number of males and females are equal in number when it comes to low annual income.

```
In [13]: fig, ax = plt.subplots(figsize=(15,8))
sns.violinplot(x = 'Gender', y = 'Annual Income (k$)', data=df, palette = 'rainbow',ax=ax)
ax.set_title('Gender vs Spending Score', fontsize = 15)
plt.show()
```

f. Correlation coefficients heatmap

```
In [44]: ## Correlation coeffecients heatmap
         fig, ax = plt.subplots(figsize=(15,8))
         sns.heatmap(df.corr(), annot=True,ax=ax).set_title('Correlation Factors Heat Map', size='15')
Out[44]: Text(0.5, 1.0, 'Correlation Factors Heat Map')
```



- The Above Graph for Showing the correlation between the different attributes of the Mall Customer Segmentation Dataset, This Heat map reflects the most correlated features with Orange Color and least correlated features with yellow color.
- We can clearly see that these attributes do not have good correlation among them, that's why we will proceed with all of the features.

5. Algorithms Clustering

We use three algorithms to clustering: ***KMean****,* ***AGNES*** *and* ***DBSCAN***

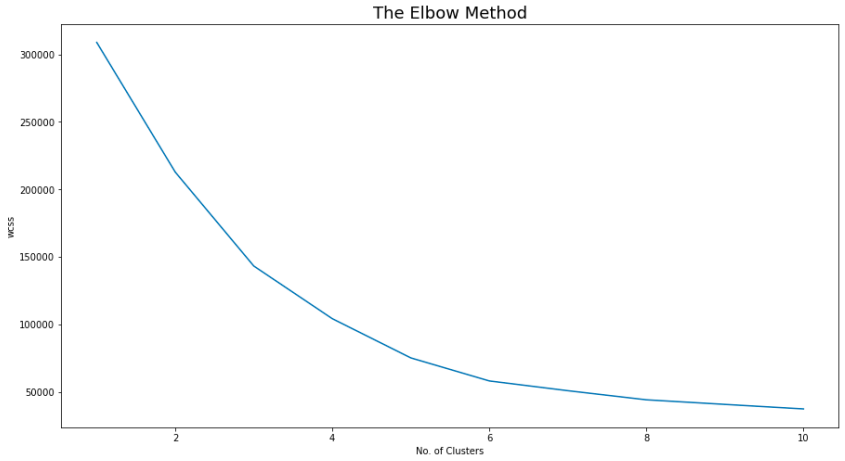
a. Kmean Algorithm

- We used library kmeans from module sklearn.cluster.

```
In [14]: from sklearn.cluster import KMeans
```

- First of all, we used the Elbow Method to identify parameter *k* for the Kmean algorithm.
  - In this step, we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k, and choose the k for which WSS first starts to diminish. In the plot of WSS-versus k, this is visible as an elbow.

In [16]:
```
wcss = []
#(Within-Cluster Sum of Square)-sum of squared distance
for i in range(1, 11):
    km = KMeans(n_clusters = i, init = 'k-means++', max_iter = 500, n_init = 10, random_state = 0)
    km.fit(x)
    wcss.append(km.inertia_)

fig, ax=plt.subplots(figsize=(15,8))
ax.plot(range(1, 11), wcss)
ax.set_title('The Elbow Method', fontsize = 18)
ax.set_xlabel('No. of Clusters')
ax.set_ylabel('wcss')
plt.show()
```

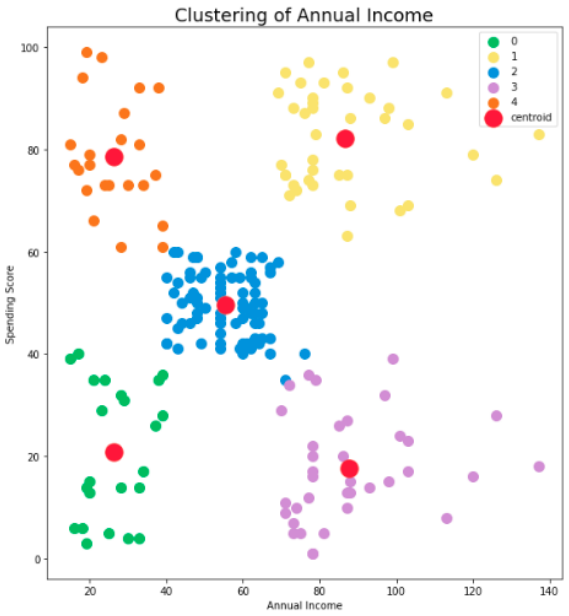


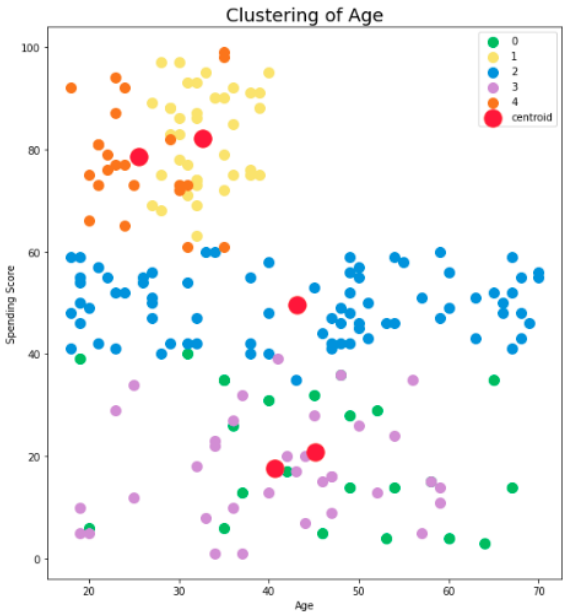
- After running this cell and analyzing the graph, we find that *k=5* is the optimal k value using the elbow method.

- After that, we run the kmean algorithm.
  - Kmeans has some parameters: n_clusters=5 is *k* that we have found above thanks to the Elbow method.
  - max_iter=300: loop the algorithm 300 times.
  - n_init=10
  - random_state=0

In [17]:
```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(20,10))

km = KMeans(n_clusters = 5, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_means = km.fit_predict(x)

ax1.scatter(x[y_means == 0, 1], x[y_means == 0, 2], s = 100, c = '#27AE60', label = '0')
ax1.scatter(x[y_means == 1, 1], x[y_means == 1, 2], s = 100, c = '#F7DC6F', label = '1')
ax1.scatter(x[y_means == 2, 1], x[y_means == 2, 2], s = 100, c = '#3498DB', label = '2')
ax1.scatter(x[y_means == 3, 1], x[y_means == 3, 2], s = 100, c = '#C39BD3', label = '3')
ax1.scatter(x[y_means == 4, 1], x[y_means == 4, 2], s = 100, c = '#E67E22', label = '4')
ax1.scatter(km.cluster_centers_[:,1], km.cluster_centers_[:, 2], s = 300, c = '#E74C3C' , label = 'centroid')

ax2.scatter(x[y_means == 0, 0], x[y_means == 0, 2], s = 100, c = '#27AE60', label = '0')
ax2.scatter(x[y_means == 1, 0], x[y_means == 1, 2], s = 100, c = '#F7DC6F', label = '1')
ax2.scatter(x[y_means == 2, 0], x[y_means == 2, 2], s = 100, c = '#3498DB', label = '2')
ax2.scatter(x[y_means == 3, 0], x[y_means == 3, 2], s = 100, c = '#C39BD3', label = '3')
ax2.scatter(x[y_means == 4, 0], x[y_means == 4, 2], s = 100, c = '#E67E22', label = '4')
ax2.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:, 2], s = 300, c = '#E74C3C' , label = 'centroid')


ax1.set_title('Clustering of Annual Income', fontsize = 18)
ax1.set_xlabel('Annual Income')
ax1.set_ylabel('Spending Score')
ax1.legend()

ax2.set_title('Clustering of Age', fontsize = 18)
ax2.set_xlabel('Age')
ax2.set_ylabel('Spending Score')
ax2.legend()

plt.show()
```

- The result of the kmeans algorithm:





- According to the result of k means algorithm, we have some **conclusions**:

This Clustering Analysis gives us a very clear insight about the different segments of the customers in the Mall.

- There are clearly five segments of Customers namely (*0: Careful Customers; 1: Target Customers; 2: General Customers; 3: Miser Customers; 4: Squandered Customers*) based on their Age, Annual Income and Spending Score which are reportedly the best factors to determine the segments of a customer in a Mall. Specifically,
    - 0: *Careful Customers* are customers that have low Annual Income and spend a few for mall shopping, and the age of them can range from 20 to 70.
    - 1: *Target Customers* are customers that spend more than 70k$ Income for shopping and spending scores of them reach above 60. The age of them is around 25-40.
    - 2: *General Customers* are the customers that are popular in the mall and they account for the large quantity. They spend about 40-70k$ income for shopping and their spending score is in the range 40-60(medium). The age of this kind of customer ranges from 20-70.
    - 3: *Miser Customers* are the customers that have high annual income but spend a low spending score for shopping. The age of them ranges from 20 to 60.
    - 4: Squandered Customers are the customers that have low annual income but spend too much for spending scores. The age of them is around 20-40.

b. Agglomerative Hierarchical Clustering Algorithm

- First of all, encoding column Gender, with 0 is Male and 1 is Female

```
gender = {'Gender': {'Male':0, 'Female':1}}
df1 = df.replace(gender)
df1.head()
```

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

- Then, normalize the data and bring all the variables to the same scale
  We use **normalize** from **sklearn.preprocessing**

```
from sklearn.preprocessing import normalize
df_scaled = normalize(df1)
df_scaled = pd.DataFrame(df_scaled, columns=df1.columns)
df_scaled.head()
```

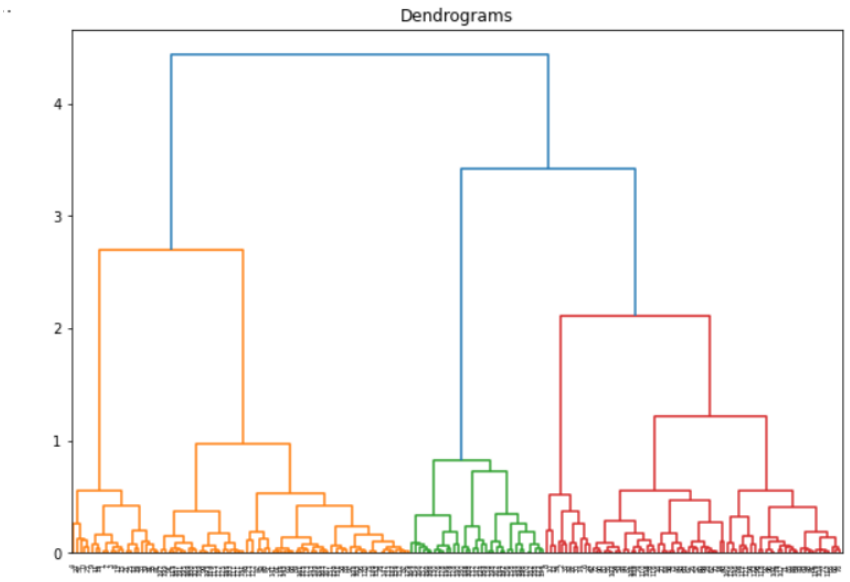
Here, we can see that the scale of all the variables is almost similar

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	0.000000	0.413925	0.326783	0.849635
1	0.000000	0.247025	0.176446	0.952809
2	0.037987	0.759737	0.607790	0.227921
3	0.012203	0.280676	0.195253	0.939653
4	0.018728	0.580581	0.318383	0.749137

** We start Agglomerative Hierarchical Clustering with all column (Gender, Age, Annual Income (k$), Spending Score (1-100))

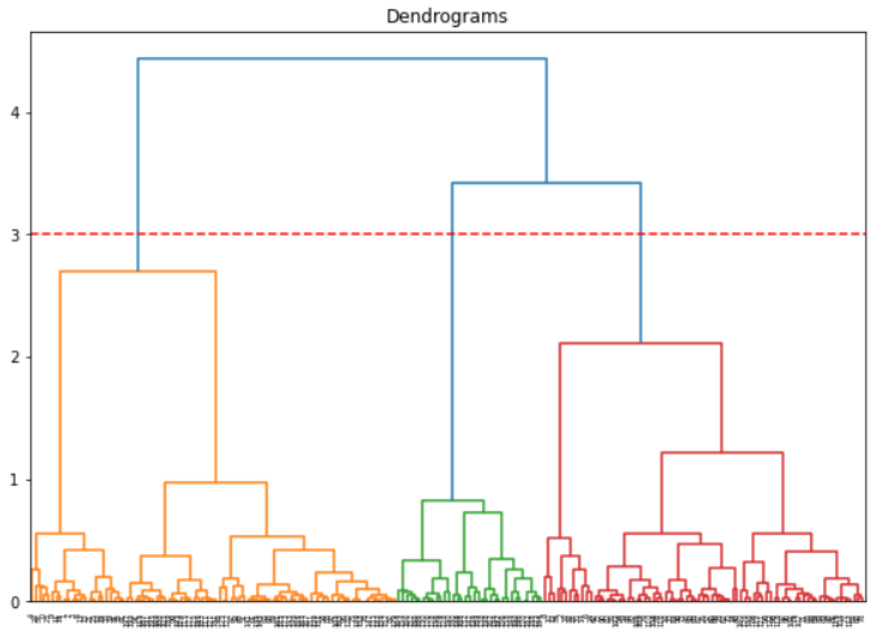
- Let's first draw the dendrogram to help decide the number of clusters for this particular problem
  + We use spicy.cluster.hierarchy to do this

```
28]
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(df_scaled, method='ward'))
```



+ The vertical line with maximum distance is the blue line and hence we can decide a threshold of 3 and cut the dendrogram

```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(df_scaled, method='ward'))
plt.axhline(y=3, color='r', linestyle='--')
```

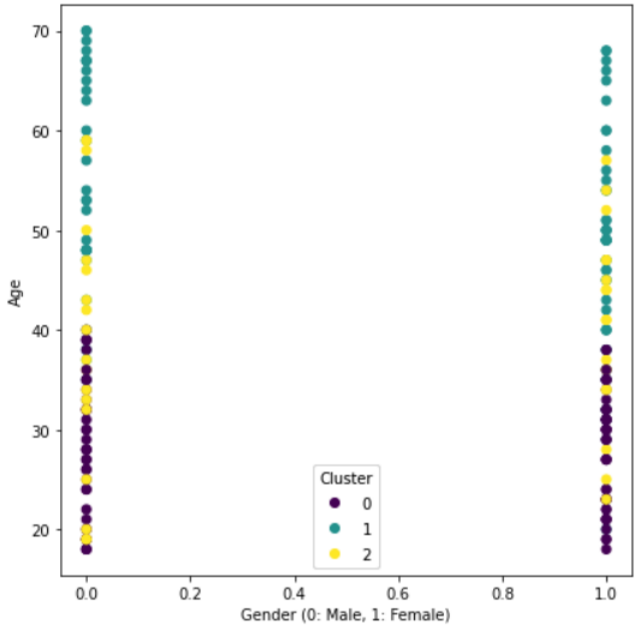


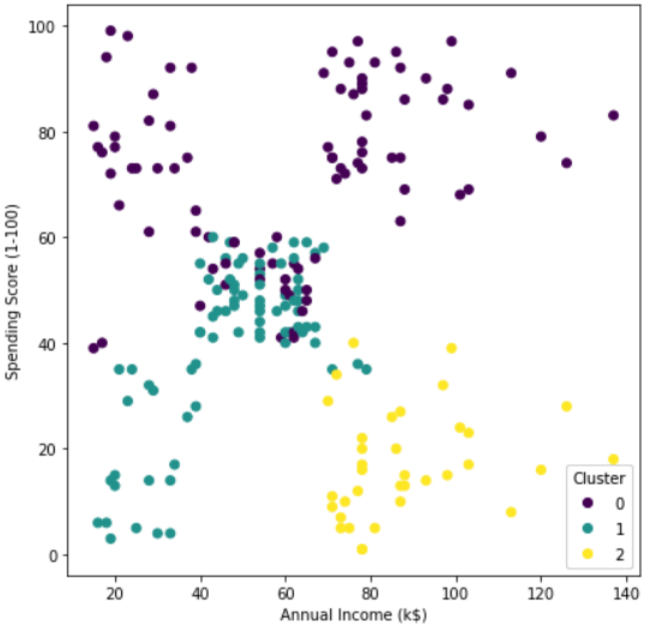
- We have three clusters as this line cuts the dendrogram at three points. Let's now apply hierarchical clustering for 3 clusters:
  + We use library **AgglomerativeClustering** from **sklearn.cluster**

```
[130]
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
cluster.fit_predict(df_scaled)
```

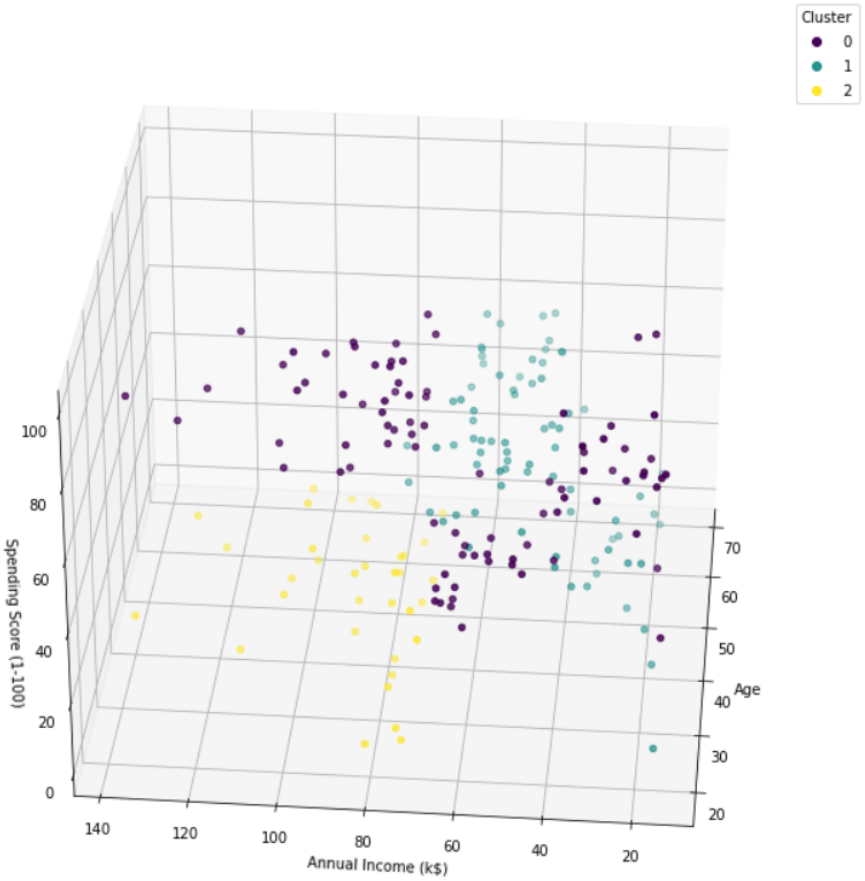
```
array([0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 2, 0, 1, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 1, 0, 2, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 1, 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])
```

  + We can see the values of 0s, 1s and 2s in the output since we defined 3 clusters. 0 represents the points that belong to the first cluster and 1 represents points in the second cluster, 2 represents points in the third cluster.
- Let's now visualize the three clusters:





  + All clusters are almost equally distributed in both Gender groups (Male, Femail)
  + Visualize 3D with 3 columns: Age, Annual Income (k$) and Spending Score (1-100)



- **Conclusions**:
  From three charts above, we can see Hierarchical Clustering algorithm generated the following **3 clusters**:
  + **Cluster 0**: Age < 40, Spending Score >= 40
  + **Cluster 1**: Age >= 40, Spending Score <= 60, Annual Income <= 80
  + **Cluster 2**: Spending Score <= 40, Annual Income > 60

c. DBSCAN algorithm

- We use library DBSCAN from **sklearn.cluster**

```
from sklearn.cluster import DBSCAN
```

- DBSCAN algorithm has two parameters:
  - Eps: the radius of the vicinity
  - MinPts: number of objects/points at least in neighboring Eps of an object

  So that we have to investigate for the best values for them
- To evaluate the eps_value and min_sample we will use Silhouette score metric

```
from sklearn.metrics import silhouette_score
```

Silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

+ 1: Means clusters are well apart from each other and clearly distinguished.

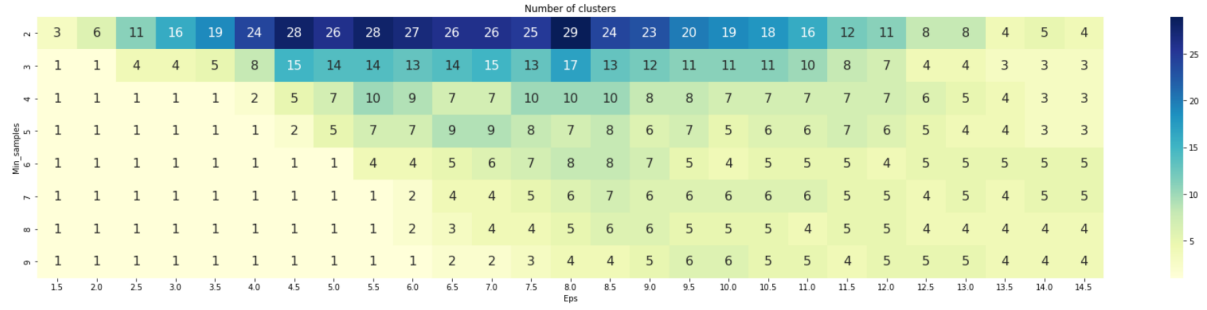
+ 0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.

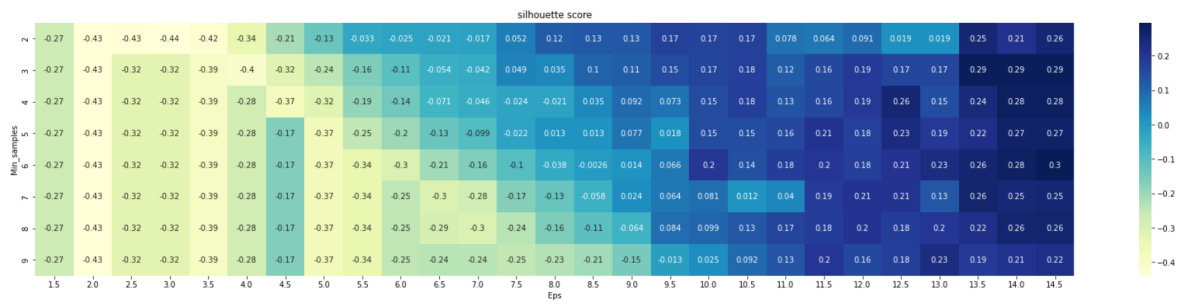
+ -1: Means clusters are assigned in the wrong way.

- First of all, we will try some **eps** and **min_samples** of DBSCAN to fit the data and find the best Silhouette score. By the way, you can also see the Number of clusters from each pair of parameters.

```
X_numerics = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
eps_values = np.arange(1.5,15,0.5) # eps values to be investigated
min_samples = np.arange(2,10) # min_samples values to be investigated
DBSCAN_params = list(product(eps_values, min_samples))

no_of_clusters = []
sil_score = []

for p in DBSCAN_params:
    DBS_clustering = DBSCAN(eps=p[0], min_samples=p[1]).fit(X_numerics)
    try:
        tmp = silhouette_score(X_numerics, DBS_clustering.labels_)
    except:
        pass
    no_of_clusters.append(len(np.unique(DBS_clustering.labels_)))
    sil_score.append(tmp)
```

- Visualize the results:





- As you can see, **min_samples** = 6 and **Eps** = 14.5 give us the best **silhouette score** = 0.3 so we will use this result for the DBSCAN algorithm.

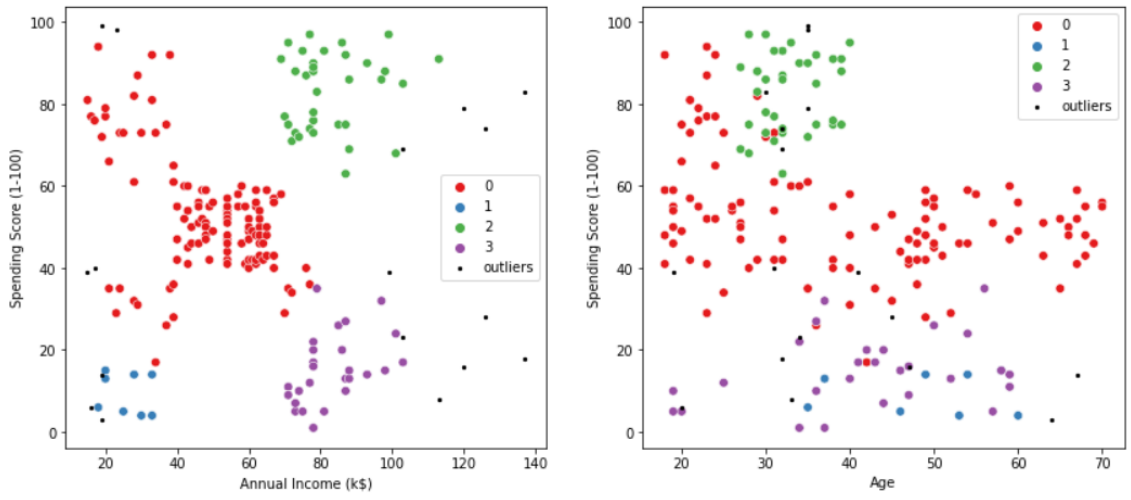
```
DBS_clustering = DBSCAN(eps=14.5, min_samples=6).fit(X_numerics)

DBSCAN_clustered = X_numerics.copy()
DBSCAN_clustered.loc[:,'Cluster'] = DBS_clustering.labels_
```

- Size of clusters

```
DBSCAN_clust_sizes = DBSCAN_clustered.groupby('Cluster').size().to_frame()
DBSCAN_clust_sizes.columns = ["DBSCAN_size"]
DBSCAN_clust_sizes
```

Cluster	DBSCAN_size
-1	17
0	113
1	8
2	35
3	27

- With those parameters, DBSCAN created 4 classes plus outliers cluster (-1). Next we will plot these data with new labels of clustering.



- **Conclusions**:

  DBSCAN algorithm generated the following 4 clusters:

  - **Class 0**: clients of all ages with medium-high spending score and medium-low annual income

- **Class 1**: Middle-age (35 - 60) clients with low spending score and low annual income
- **Class 2**: Younger clients (25 - 40) with high spending score and medium annual income
- **Class 3**: Below 60 years old with low spending score and medium annual income

6. Self-assessment of the team's work results

- Each member of the group has completed the assigned tasks, namely, each person has researched and implemented different clustering functions.
- Every job is meticulously completed by each member of the team.
- Therefore, it seems to us that we get a high score in this project.

7. Reference

[1]https://towardsdatascience.com/customer-segmentation-using-k-means-clustering-d33964f238c3
[2]https://www.kaggle.com/code/datark1/customers-clustering-k-means-dbscan-and-ap
[3]https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c
[4] Hierarchical Clustering