Cluster Analysis on Customer Segmentation

**Under the Guidance of
Professor Gary Davis**

Bharath Anand
#ID: 02044023
Graduate Student
Department of Data Science
University of Massachusetts Dartmouth

Balakrishna Vardhineni
#ID: 02069565
Graduate Student
Department of Data Science
University of Massachusetts Dartmouth

Cluster Analysis on Customer Segmentation

1. Abstract

To evaluate customer data and find common patterns and qualities among a group of consumers, unsupervised learning is a form of machine learning technique that can be utilized. In this project, we performed testing on the two most common unsupervised learning algorithms K-Means and Hierarchical clustering techniques.

2. Introduction and Background

For this report, we've used the Mall customer dataset to perform cluster analysis on customer segmentation and the dataset file can be found on the Kaggle website. The dataset can be downloaded from here.

In any firm, it is crucial to identify clients based on their preferences and other actions. Using this identification may make reaching customers with particular offers and goods easier. A company with a high number of customers can find it challenging to recognize and record each one separately. Companies can build customer groups with distinct traits by utilizing clustering algorithms to group like consumers together. This enables them to better target each segment with their marketing and sales initiatives. The process of concluding the vast amounts of client information that have been gathered involves a great deal of data processing and automated methods.

Clustering, a machine learning (ML) unsupervised technique, aids in client identification based on their essential traits. Unsupervised learning algorithms are effective tools for businesses wanting to better understand and service their consumers since they can be taught vast amounts of data. The identification and segmentation of customers using the K-Means and hierarchical clustering algorithms will be covered in this article. When we use these Python implementations, the clustering result will be visible. Finally, we will compare the effectiveness of K-Means and hierarchical clustering, two clustering methodologies.

Consider a mall that has 200 customer records on file thanks to a membership drive. It now has data on its clients, such as their gender, age, annual income, and spending score. Customers are assigned a spending score based on their previous mall-related purchases and spending patterns.

Now imagine that the mall is introducing a high-end product and is looking to connect with potential buyers. It will take a long time and be expensive to approach every customer. As a result, the mall might decide to separate customers based on their potential to purchase a luxury good. Clustering can be used to solve this issue because it allows us to categorize clients into distinct groups and locate potential customers.

The client can be represented as a two-dimensional Euclidean space, with the X-axis representing the customer's annual revenue and the Y-axis representing the customer's spending score. Using a clustering technique, we may create groups of customers after representing each consumer on this plane. We can now pinpoint the set of clients who can purchase a luxury good based on their income and spending score.

We used the clustering techniques mentioned above in our project for the analysis of customer segmentation from the mall customer dataset. In this report, we tried to achieve the analysis and visual representation of the clustering techniques discussed above.

2. Clustering Methods

2.1. K-means Clustering

Within the unlabeled dataset, intrinsic groups are found using K-Means clustering, and conclusions are drawn from them. Centroid-based clustering serves as its foundation.

Centroid: A data point at the center of a cluster is known as the centroid. A centroid is used to represent clusters in centroid-based clustering. The idea of similarity is determined by how near a data point is to the cluster centroid in this iterative procedure. The K-Means clustering algorithm uses an iterative process to produce a final result. It operates as follows. The data set and the number of clusters K inputs for the method. Each data point's features are included in the data set. Initial K centroids estimates form the basis of the method.

1. **Data assignment step:** Each centroid defines one of the clusters. Here, based on the squared Euclidean distance, each data point is matched to its nearest centroid. Therefore, each data point is assigned to a cluster based on the smallest Euclidean distance if ci is the collection of centroids in set C.
2. **Centroid Update step:** The centroids are updated and recomputed in this stage. Calculating the average of all the data points assigned to that centroid's cluster is accomplished. After that, the algorithm repeats steps 1 and 2 until a stopping criterion is satisfied. The minimization of the sum of the distances, the absence of data points changing the clusters, or a certain maximum number of iterations is all examples of stopping criteria. This algorithm will always lead to a conclusion. The answer might be a local optimum, which means that examining more than one algorithm run with randomly chosen starting centroids may produce a better result.

Choosing the value of K:

Finding the number of clusters and data labels for a given value of K is essential to the K-Means algorithm. We must execute the K-Means clustering technique for various values of K and compare the results to determine the number of clusters in the data. K's value determines how well the K-Means algorithm performs. We should select the K number that will offer us the best performance. The ideal value of K can be found using a variety of methods. The elbow method, which is explained here, is the most popular technique.

The Elbow Method:

The ideal number of clusters for K-means clustering is chosen using the elbow approach. The value of the cost function generated by various K values is plotted using the elbow approach.

Average distortion will go down when K goes up. The constituent instances in each cluster will thereafter be less numerous and more closely spaced from their respective centroids. The reduction in average distortion will, however, occur as K rises. The elbow, or the point at which we should stop further clustering the data, is the value of K at which improvement in distortion drops the most.

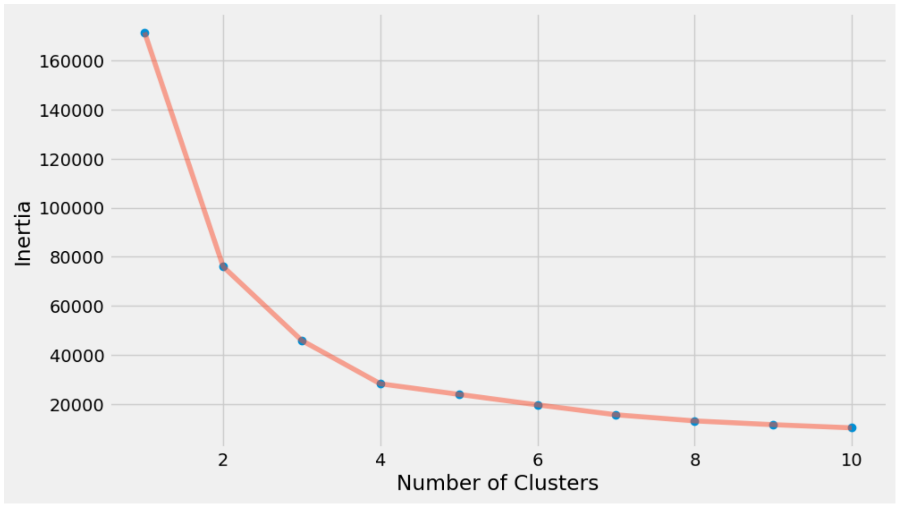


Fig 1. Elbow method for choosing 'k'

In our case, the value of k is 5, it's the optimum number of clusters to group the customers.

2.2. Hierarchical Clustering

In this project, hierarchical clustering will be used. Agglomerative clustering is the most used hierarchical clustering technique used to group things based on how similar they are to one another. It also goes by the name AGNES (Agglomerative Nesting). The program first treats each object as a singleton cluster. Pairs of clusters are gradually combined when all clusters have been merged into a single giant cluster that contains all elements. A dendrogram, a tree-based representation of the objects, is the result. This could help with focused marketing and communication efforts to particular consumer groups as well as the identification of various customer personality types.

Unlike the K-Means algorithm, hierarchical clustering doesn't require getting clusters at the beginning itself and cluster size can be varied. To get the number of clusters for hierarchical clustering, we make use of a Dendrogram. A dendrogram is a tree-like diagram that records the sequences of merges or splits. Whenever we merge two clusters, a dendrogram will record the distance between these clusters and represent it in graph form.

3. Results

The cluster formed using K-means clustering using k = 5.

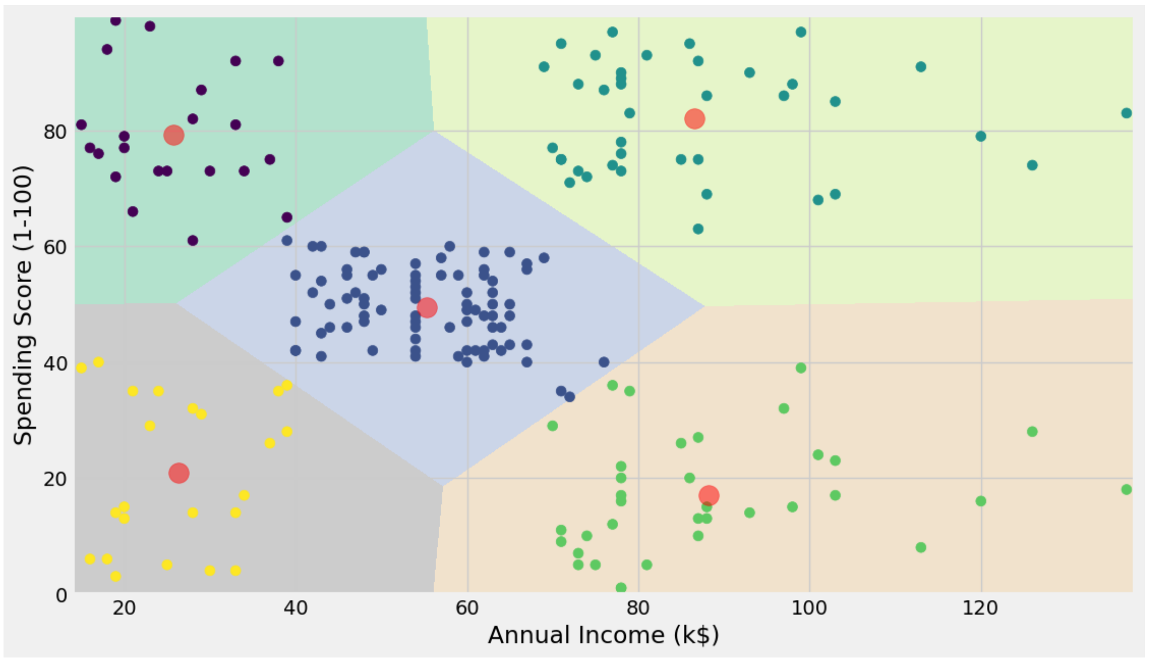


Fig 2. The cluster formed using K-Means Algorithm when k=5

The 3-Dimensional cluster view of the cluster formed using K-Means Algorithm

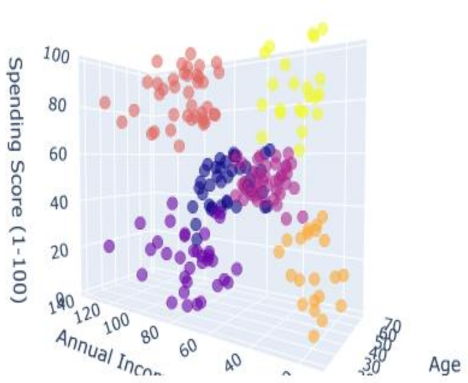


Fig 3. 3-Dimensional cluster view

Dendrogram formed using Hierarchical clustering

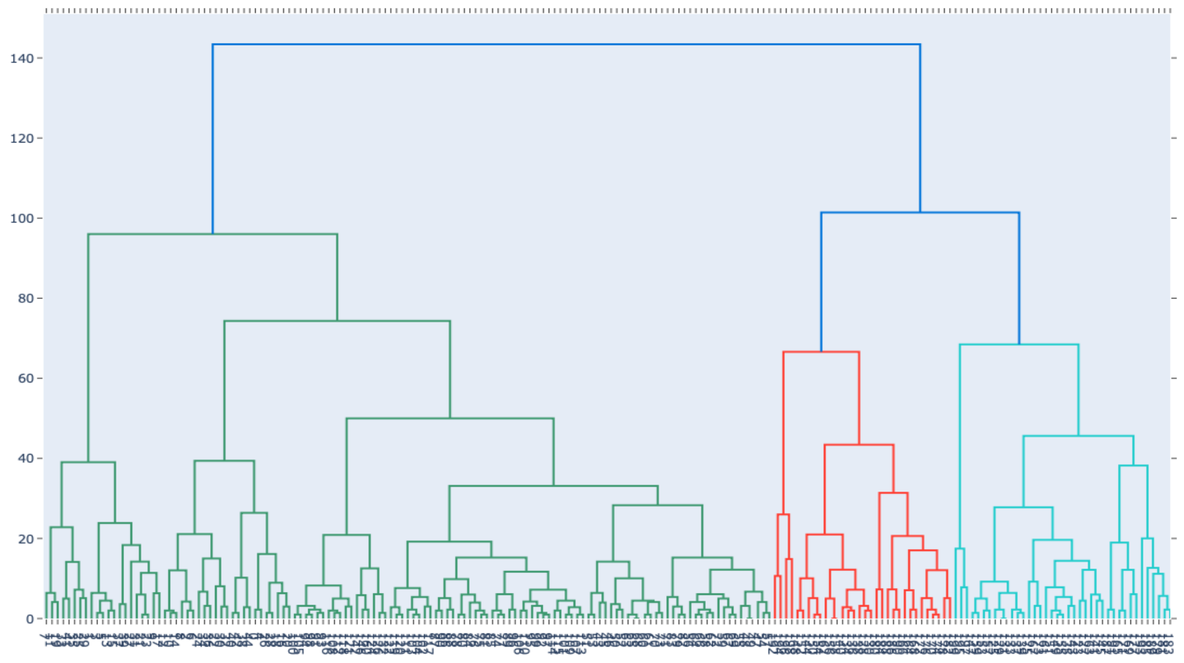


Fig 4. Dendrogram for Customer segmentation

Dendrogram formed using Hierarchical clustering with the threshold value

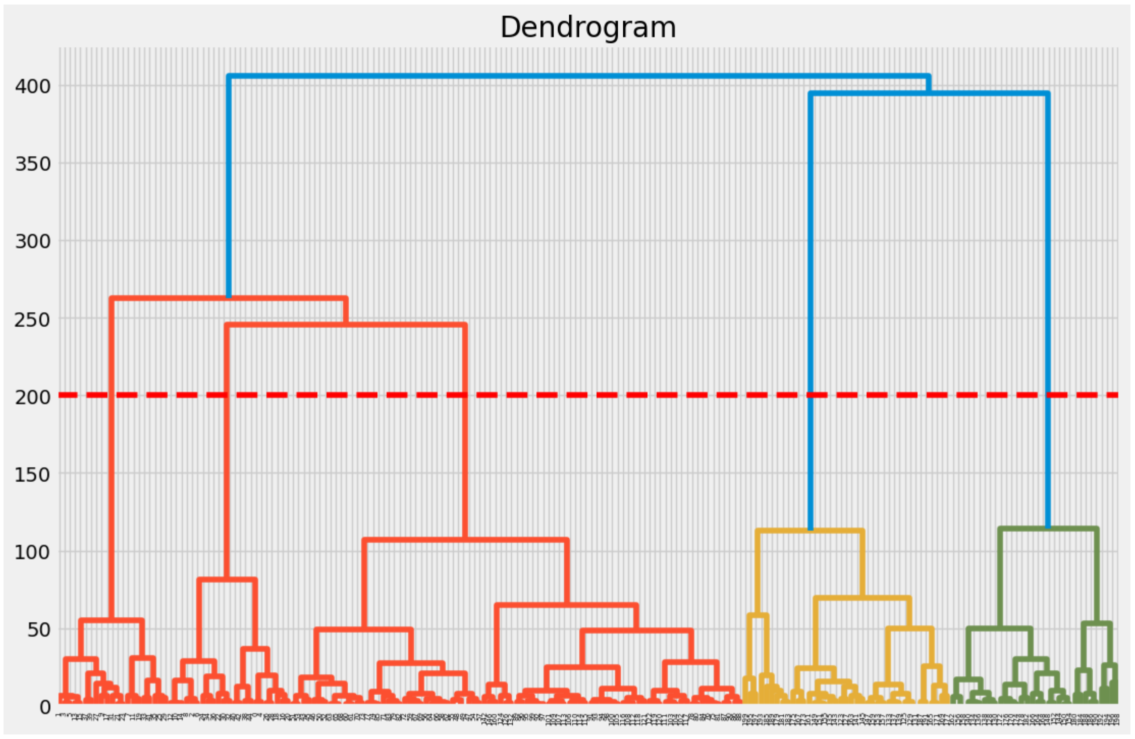


Fig 5. Dendrogram for Customer segmentation with threshold value

The formed cluster using the Hierarchical technique:

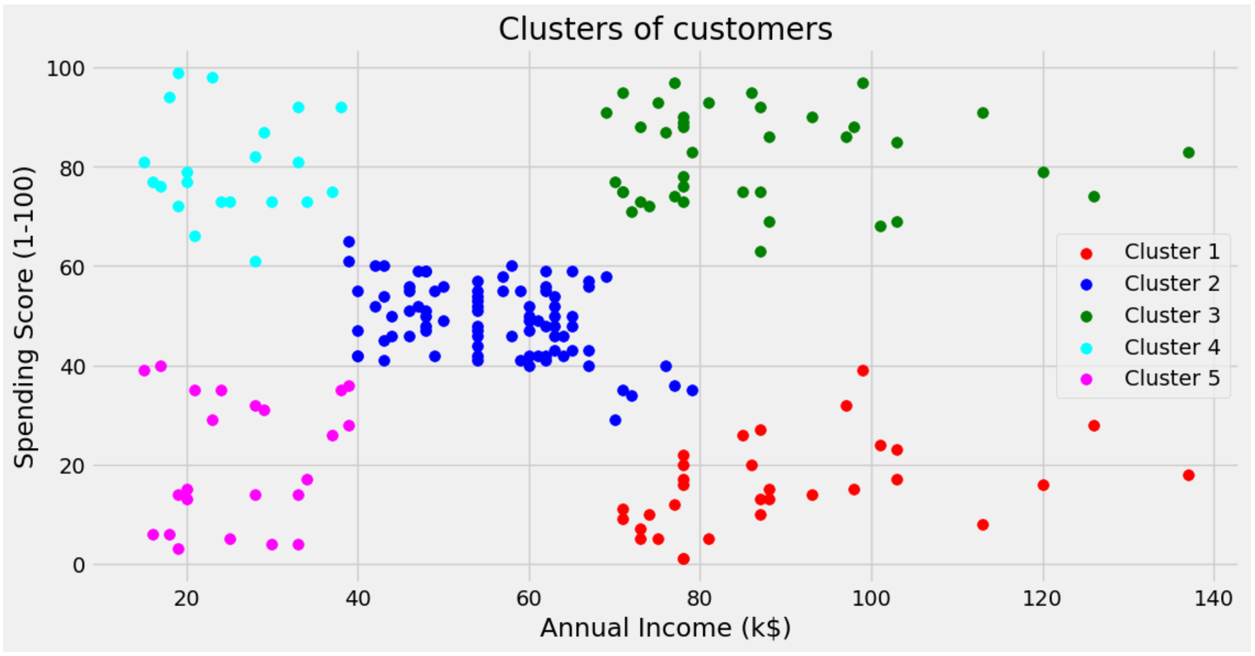


Fig 6. The cluster formed using Hierarchical technique

More details of the report can be found in the attached appendix python notebook with the code review. We did most of the detailed report in the markdown file.

4. Conclusion and Discussions:

By performing the data analysis, processing and visualization, this dataset was clustered in very efficient ways. Regarding the age vs spending score, we see that it's a bit unclear how the data could be clustered, so the algorithms help us a lot. By using the above two unsupervised learning machine-learning techniques we were able to get the cluster analysis of customer segmentation.

In K-Means clustering analysis, we initially performed segmentation using Age and spending scores and generated 4 clusters of visual representation with centroids since the number of clusters near an elbow of 4 from inertia vs cluster number graph. Next, performed segmentation using Annual income and spending scores and generated 5 clusters of visual representation. Finally, we presented a 3D visual representation of Age, Spending score, and Annual Income. From the results, different clusters formed based on customer segmentation. It has shown that customer variance is based on the spending score, Annual income, and ages of customers.

In Hierarchical clustering analysis, initially, a dendrogram was formed between the Annual Income and Spending score and found the threshold value of clusters i.e., 5. Next, we performed a visualization of clusters between the Spending score and Annual Income. From the visuals, we got the results of customer variance by spending score and annual income.

When comparing the clustering techniques used in the analysis, the convergence is guaranteed in K-Means and it worked well for the clustering analysis of customer segmentation with multiple samples from the dataset. It was easy to conclude the results from K-Means clustering visual representations. Finding out the best set of clusters was difficult in K-Means; However, we were able to get it from the elbow of the Inertia vs cluster count graph. With the Hierarchical clustering, the calculation of cluster value was easier compared to K-Means. But the running time is very high compared to K-Means.

From the results, it is concluded that it’s not easy to predict the spending capability of a customer based on age and annual income from the dataset we selected. However, we were able to predict some of the cluster analysis efficiently as possible, as the customers with high income have a high spending score compared to low annual income customers.

The challenges we faced during this project were mostly about insufficient data and also not much association between the other factors of the dataset. The accuracy and reliability of the Mall customer segmentation predictions using unsupervised learning algorithms are highly dependent on the quality and quantity of data available for analysis.

Next, we’ll focus on a larger dataset selection and work on improving the performance of clustering techniques by applying different methods to predict clusters’ values. We’ll try to work on different techniques of cluster analysis based on the type of dataset we selected.

References:

1. Most Popular Clustering Algorithms Used in Machine Learning.

   URL: https://analyticsindiamag.com/most-popular-clustering-algorithms-used-in-machine-learning/.

2. Clustering Techniques Every Data Science Beginner Should Swear By.

   URL: https://analyticsindiamag.com/clustering-techniques-every-data-science-beginner-should-swear-by/.

3. Margareta Ackerman and Shai Ben-David (2016). A characterization of linkage-based hierarchical clustering. Journal of Machine Learning Research, 17:1–17, 2016. URL: https://www.jmlr.org/papers/volume17/11-198/11-198.pdf

Other References:

- Python 3: https://www.python.org/.
- Pandas: https://pandas.pydata.org/.
- NumPy: https://numpy.org/.
- Seaborn: https://seaborn.pydata.org/.
- Matplotlib: https://matplotlib.org/.
- K-Means Clustering: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.
- Hierarchical Clustering: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

Appendix

The detailed review of jupyter workbook attached below along with the comments and observations we did from graphs and visualization of clustering techniques.

Customer Segmentation and Analysis using K-means and Hierarchical Clustering Techniques

Importing Libraries

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as py
import plotly.graph_objs as go from
sklearn.cluster import KMeans import
warnings
import os
warnings.filterwarnings("ignore")
py.offline.init_notebook_mode(connected = True)
```

Data Exploration

In [2]:

```
df = pd.read_csv('./Mall_Customers.csv.xls') df.head()
```

Out[2]:

	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 [3]:

```
df.shape
```

Out[3]:

```
(200, 5)
```

In [4]:

```
df.describe()
```

Out[4]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

In [5]:

```
df.dtypes
```

Out[5]:

```
CustomerID                        int64
Gender                           object
Age                               int64
Annual  Income  (k$)  int64  Spending
Score (1-100) int64 dtype: object
```

Check for missing values in dataset

In [6]:

```
df.isnull().sum()
```

Out[6]:

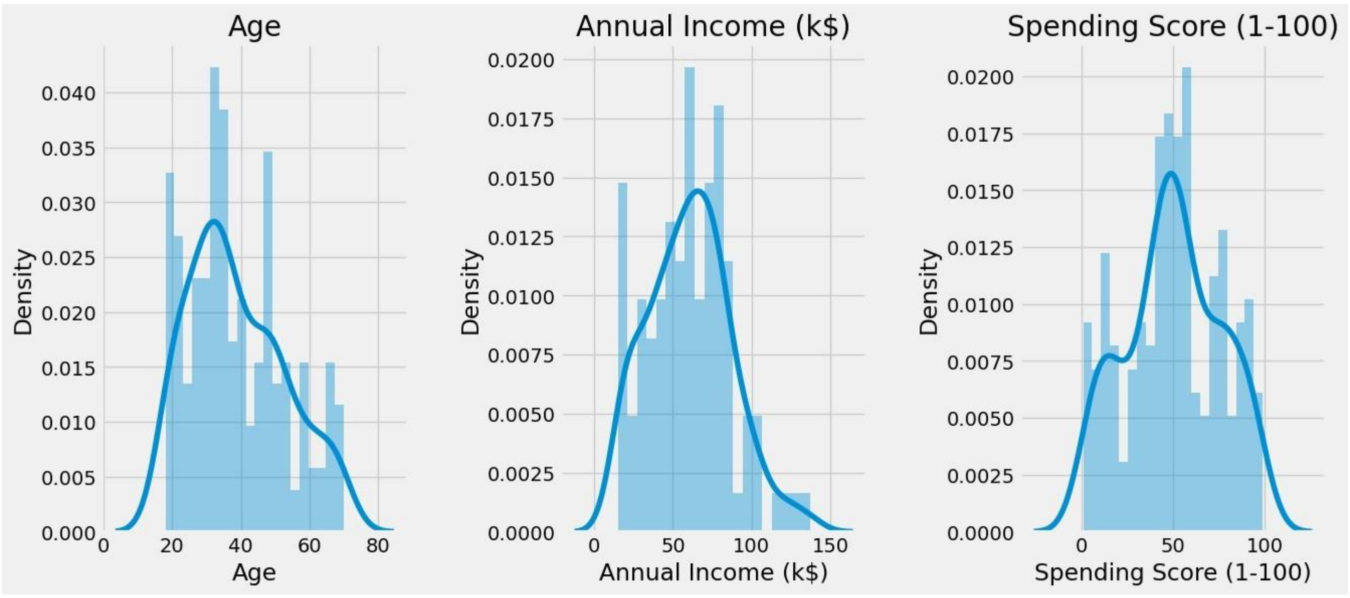
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Data Visualization

In [7]:

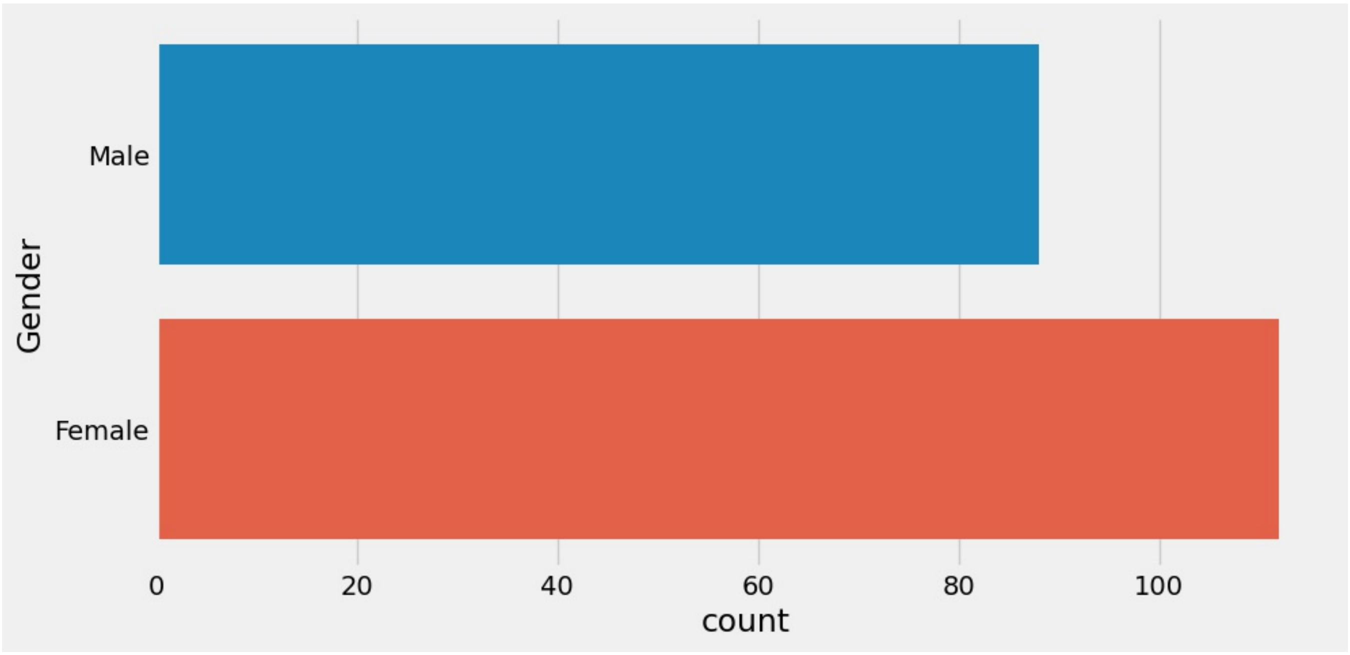
```
plt.style.use('fivethirtyeight')
```

```
In [8]: plt.figure(1 , figsize = (14 , 6)) n = 0
        for x in ['Age' , 'Annual Income (k$)' , 'Spending Score(1-100)']: n += 1
              plt.subplot(1 , 3 , n) plt.subplots_adjust(hspace =0.5 ,
              wspace = 0.5) sns.distplot(df[x] , bins = 20)
              plt.title('{}'.format(x))
        plt.show()
```



Count plot of Gender

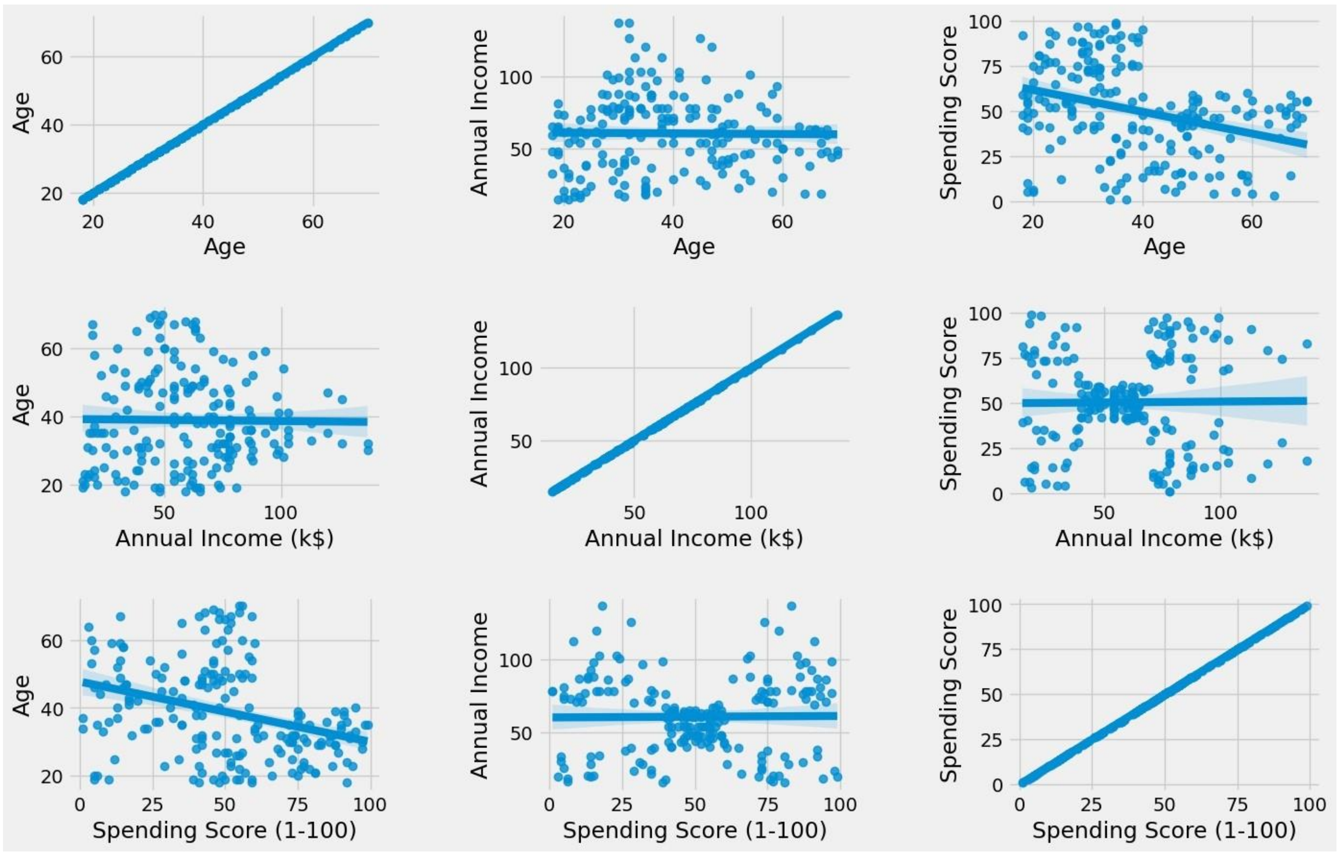
```
In [9]: plt.figure(1 , figsize = (10 , 5)) sns.countplot(y =
        'Gender' , data = df) plt.show()
```



We can see that Female count is greater than male count from data.

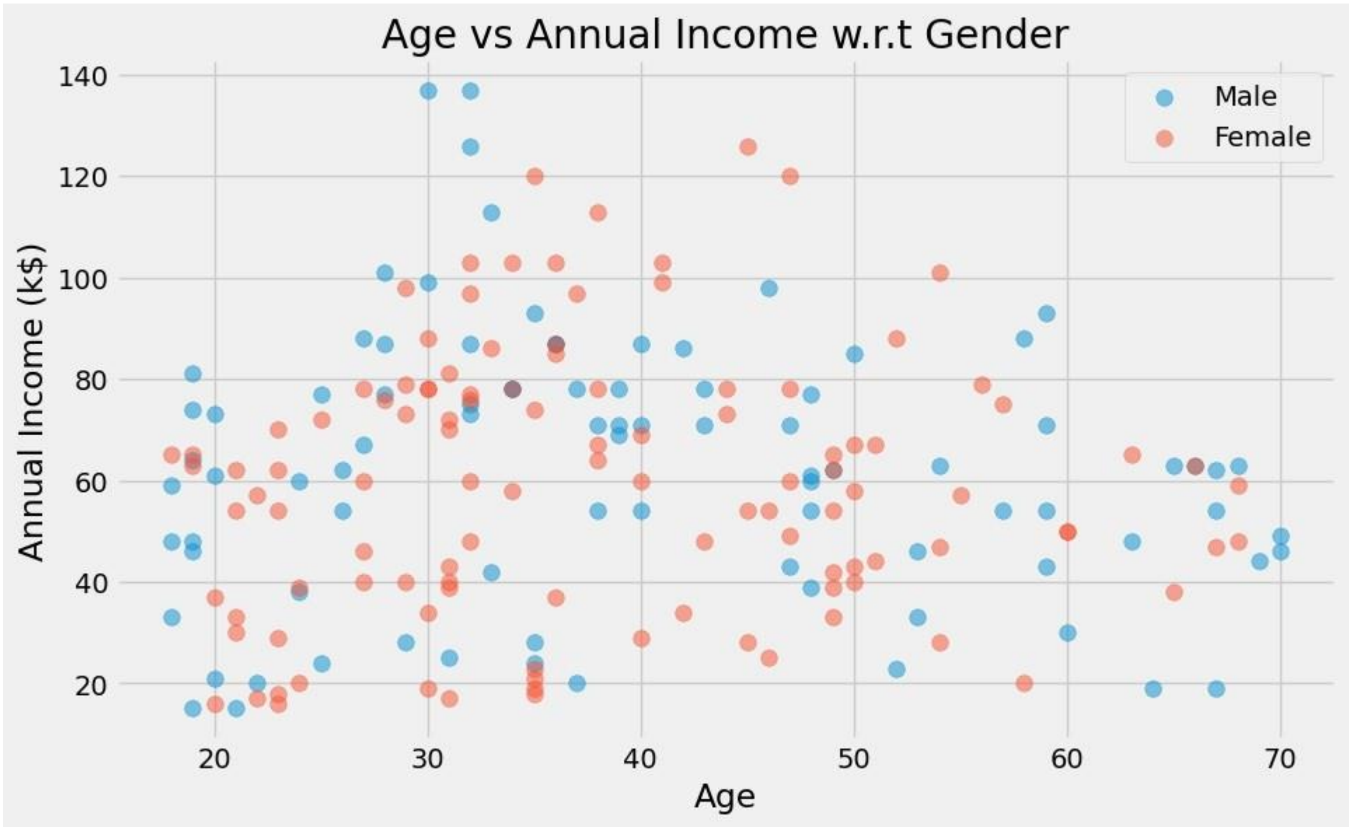
Ploting the Relation between Age , Annual Income and Spending Score

```
In [10]: plt.figure(1 , figsize = (15 , 10)) n = 0
         for x in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
               for y in ['Age' , 'Annual Income (k$)' , 'Spending Score(1-100)']: n += 1
                     plt.subplot(3 , 3 , n) plt.subplots_adjust(hspace = 0.5 ,
                     wspace =0.5) sns.regplot(x = x , y = y , data = df)
                     plt.ylabel(y.split()[0]+' '+y.split()[1] if len(y.split()) > 1 else y) plt.show()
```



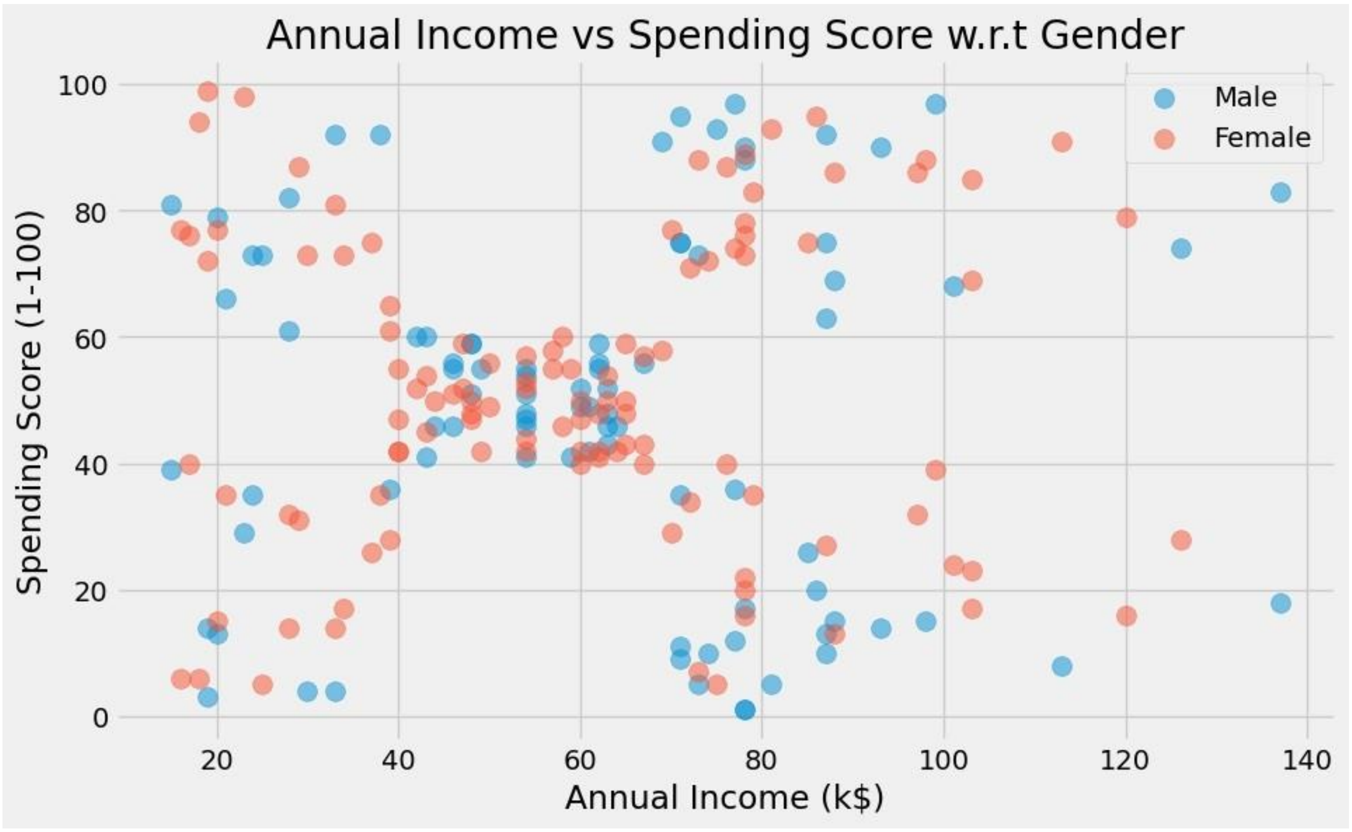
Plot between Age and Annual Income w.r.to Gender

In [11]:
```
plt.figure(1 , figsize = (10 ,6))
for gender in ['Male' , 'Female']:
    plt.scatter(x = 'Age' , y = 'Annual Income (k$)' , data = df[df['Gender'] == gender] , s = 70 , alpha
                = 0.5 , label = gender)
plt.xlabel('Age'), plt.ylabel('Annual Income (k$)')
plt.title('Age vs Annual Income w.r.t Gender') plt.legend()
plt.show()
```



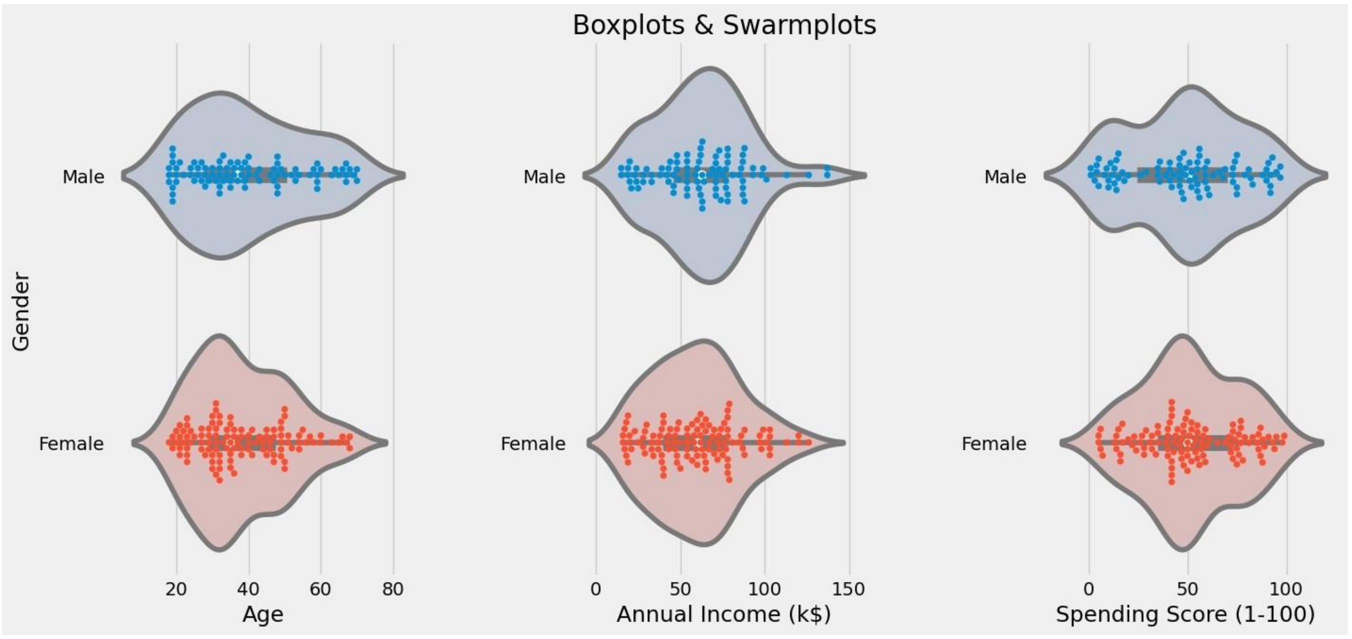
Plot between Annual Income and Spending score w.r.to Gender

In [12]:
```
plt.figure(1 , figsize = (10 ,6))
for gender in ['Male' , 'Female']:
    plt.scatter(x = 'Annual Income (k$)',y = 'Spending Score (1-100)' ,
                    data = df[df['Gender'] == gender] ,s = 100 , alpha = 0.5 , label =gender) plt.xlabel('Annual Income (k$)'),
plt.ylabel('Spending Score (1-100)')
plt.title('Annual Income vs Spending Score w.r.t Gender') plt.legend()
plt.show()
```



Distribution of values in Age , Annual Income and Spending Score according to Gender

In [13]:
```
plt.figure(1 , figsize = (15 , 7)) n = 0
for cols in ['Age' , 'Annual Income (k$)' , 'Spending Score(1-100)']: n += 1
      plt.subplot(1 , 3 , n) plt.subplots_adjust(hspace = 0.5 ,
      wspace =0.5)
      sns.violinplot(x = cols , y = 'Gender' , data = df , palette ='vlag') sns.swarmplot(x
      = cols , y = 'Gender' , data = df) plt.ylabel('Gender' if n == 1 else '')
      plt.title('Boxplots & Swarmplots' if n == 2 else '') plt.show()
```



Clustering using K-means technique

Firstly, let's assume that Age and Spending Score columns could form clusters together, so let's just apply the fitting function at the class sklearn.Kmeans. The purpose here is to obtain as many values of cluster inertia as possible, selecting the closest value to the elbow of the inertia X cluster_number graph.

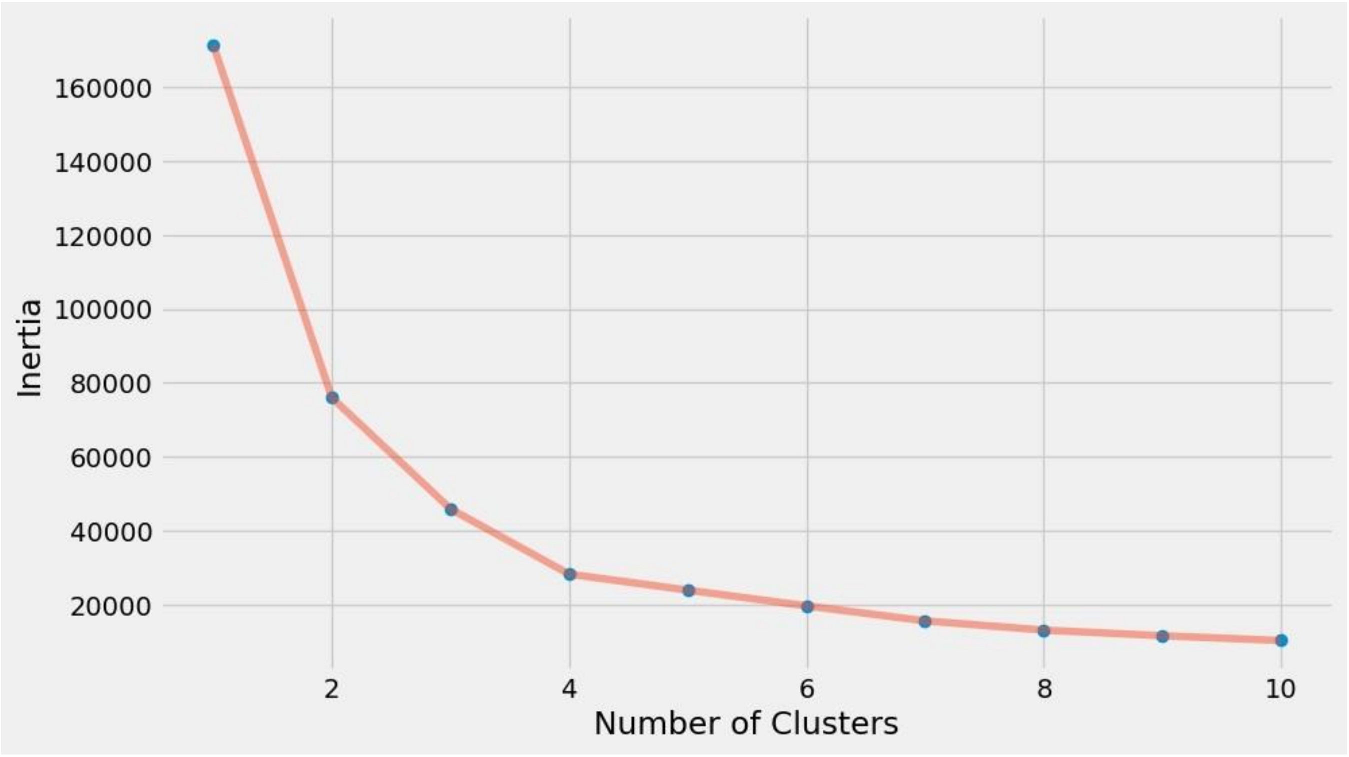
Based on the best cluster size, the model is updated and the labels are associated with each row on the dataset.

1. Segmentation using Age and Spending Score

In [14]:
```
'''Age and spending Score'''
X1 = df[['Age' , 'Spending Score (1-100)']].iloc[: ,:].values inertia = []
for n in range(1 , 11):
      algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001,
                                    random_state= 111 , algorithm='elkan') )
      algorithm.fit(X1) inertia.append(algorithm.inertia_)
```

Selecting N Clusters based in Inertia (Squared Distance between Centroids and data points, should be less)

In [15]:
```
plt.figure(1 , figsize = (10 ,6)) plt.plot(np.arange(1 , 11) ,
inertia , 'o')
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5) plt.xlabel('Number of
Clusters') , plt.ylabel('Inertia') plt.show()
```

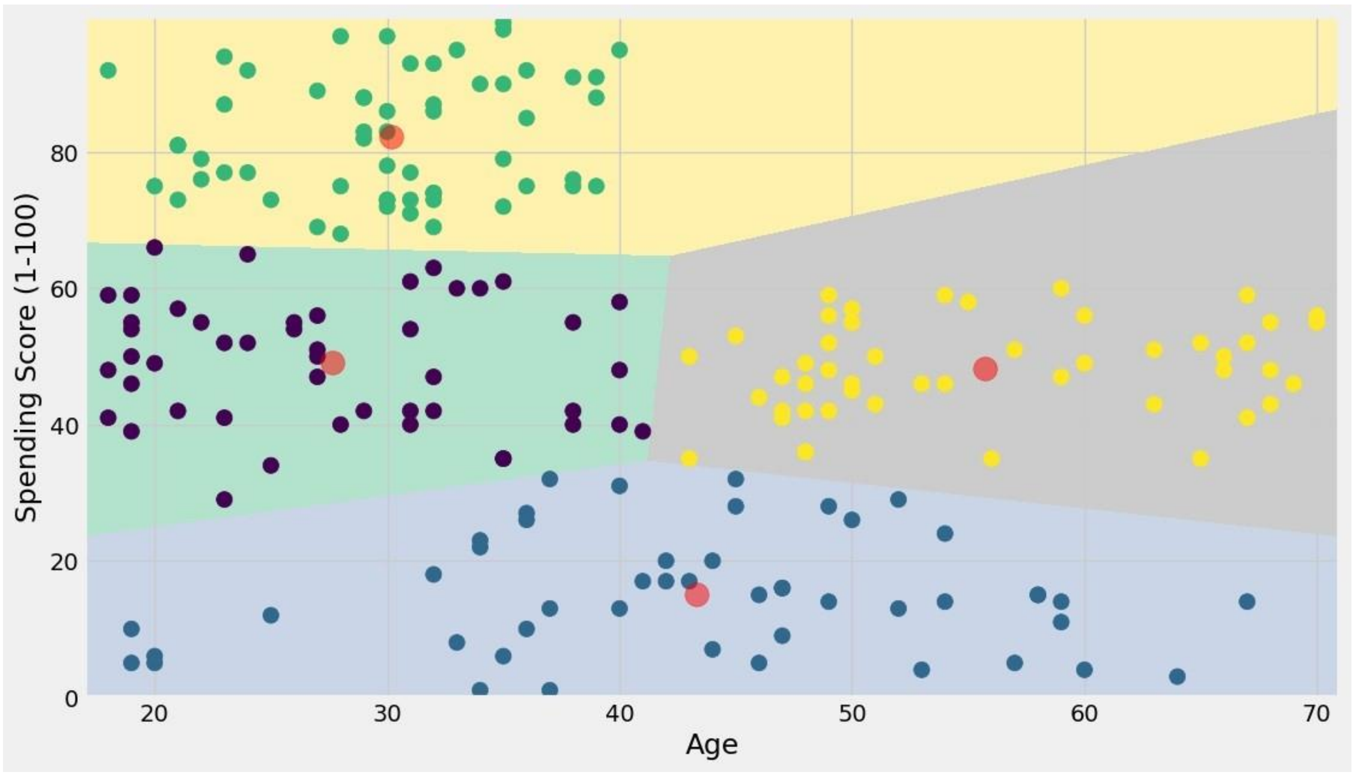


Considerng 4 as cluster size from the plot as it's near to elbow.

In [16]:
```
algorithm = (KMeans(n_clusters = 4 ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001,
                                    random_state= 111 , algorithm='elkan') )
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_
```

In [17]:
```
h = 0.02
x_min, x_max = X1[:, 0].min() - 1, X1[:, 0].max() +1
y_min, y_max = X1[:, 1].min() - 1, X1[:, 1].max() +1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,h)) Z =
algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [18]:
```
plt.figure(1 , figsize = (12 , 7) ) plt.clf()
Z = Z.reshape(xx.shape)
plt.imshow(Z , interpolation='nearest',
                extent=(xx.min(), xx.max(), yy.min(), yy.max()),
                cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Age' ,y = 'Spending Score (1-100)' , data = df , c = labels1, s = 90 )
plt.scatter(x = centroids1[: , 0] , y = centroids1[: , 1] , s = 200 , c = 'red' , alpha = 0.5) plt.ylabel('Spending Score (1-100)') ,
plt.xlabel('Age')
plt.show()
```

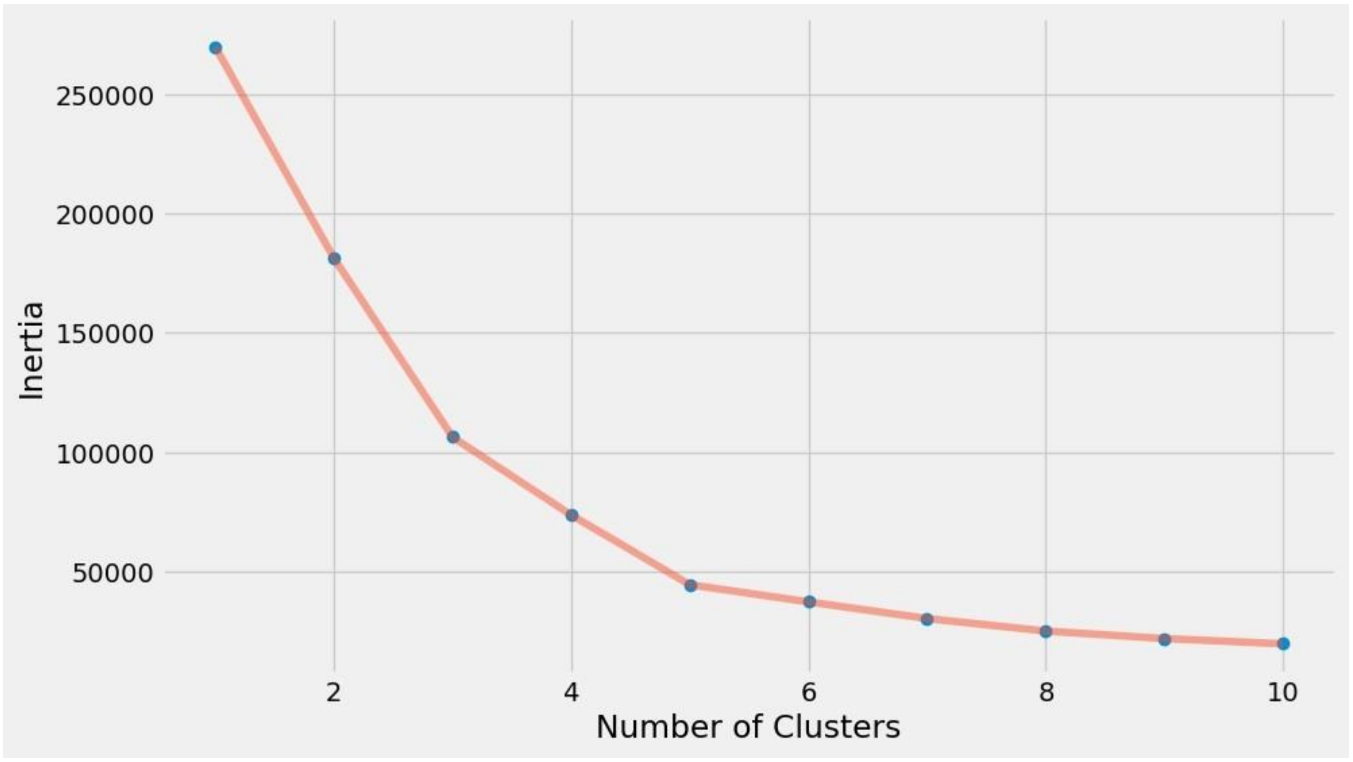


Now, let's assume that Annual Income and Spending Score columns could form clusters together, so let's just apply the fitting function at the class sklearn.Kmeans.

2. Segmentation using Annual Income and Spending Score

In [19]:
```
'''Annual Income and spending Score'''
X2 = df[['Annual Income (k$)' , 'Spending Score (1-100)']].iloc[: ,:].values inertia = []
for n in range(1 , 11):
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001,
                        random_state= 111 , algorithm='elkan') )
    algorithm.fit(X2) inertia.append(algorithm.inertia_)
```

In [20]:
```
plt.figure(1 , figsize = (10 ,6)) plt.plot(np.arange(1 , 11) ,
inertia , 'o')
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5) plt.xlabel('Number of
Clusters') , plt.ylabel('Inertia') plt.show()
```

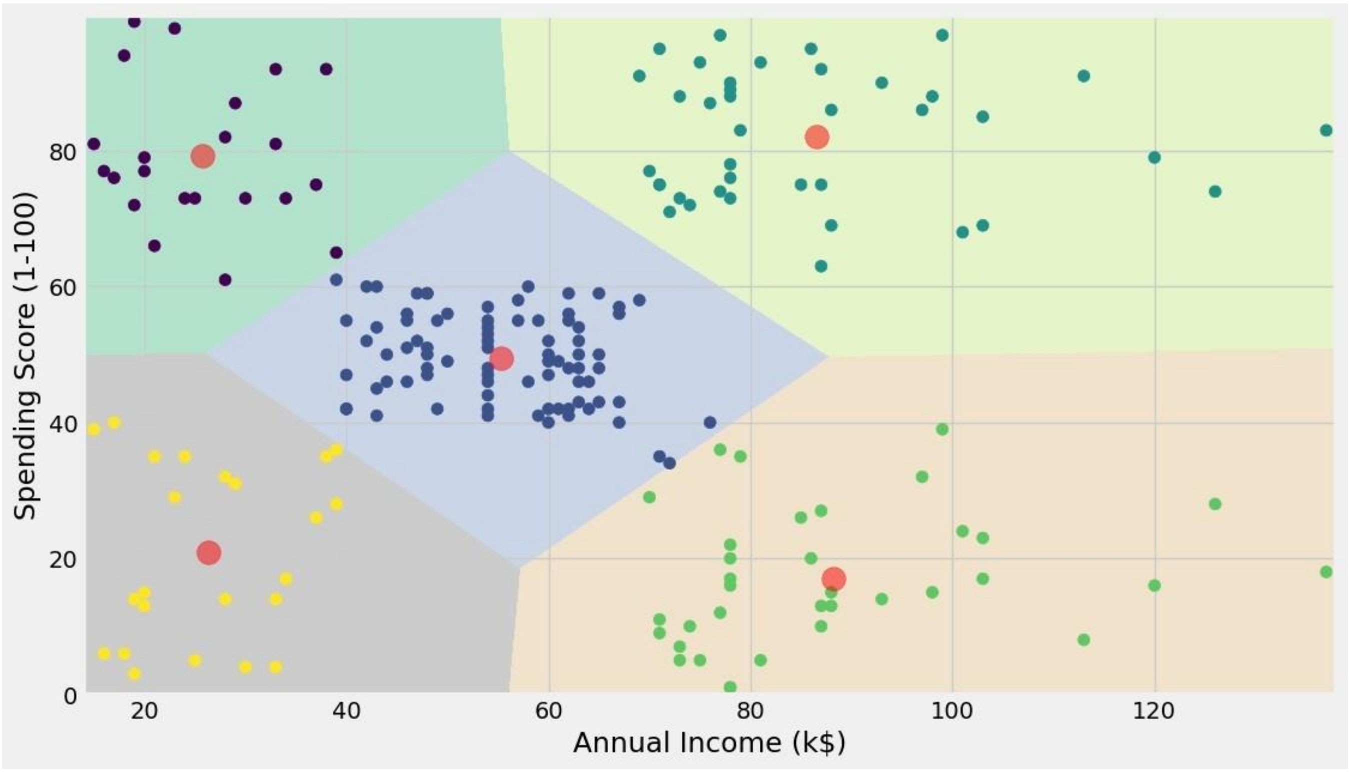


Considering 5 as number of clusters as it's near to elbow.

In [21]:
```
algorithm = (KMeans(n_clusters = 5 ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001,
                        random_state= 111 , algorithm='elkan') )
algorithm.fit(X2)
labels2 = algorithm.labels_
centroids2 = algorithm.cluster_centers_
```

In [22]:
```
h = 0.02
x_min, x_max = X2[:, 0].min() - 1, X2[:, 0].max() +1
y_min, y_max = X2[:, 1].min() - 1, X2[:, 1].max() +1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,h)) Z2 =
algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [23]:
```
plt.figure(1 , figsize = (12 , 7) ) plt.clf()
Z2 = Z2.reshape(xx.shape)
plt.imshow(Z2 , interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Annual Income (k$)' ,y = 'Spending Score (1-100)' , data = df , c = labels2 , s = 50 )
plt.scatter(x = centroids2[: , 0] , y = centroids2[: , 1] , s = 200 , c = 'red' , alpha = 0.5) plt.ylabel('Spending Score (1-100)') ,
plt.xlabel('Annual Income (k$)')
plt.show()
```



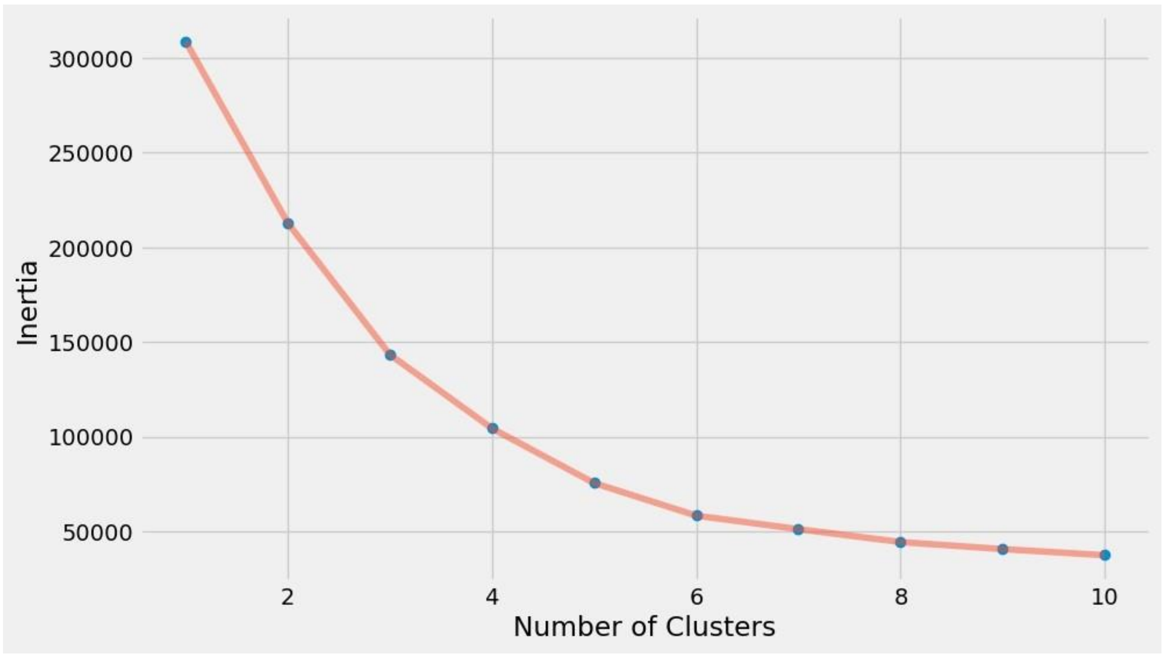
3. Segmentation using Age, Annual Income and Spending Score

Here, we can apply multi-dimensional K-Means Clustering Analysis. The curve of inertia here is harder than the previous ones to interpret, but a visual inspection at the plot is of help in these situations.

The final result can be seen at the second graph below.

In [24]:
```
X3 = df[['Age' , 'Spending Score (1-100)' ,'Annual Income (k$)']].iloc[: , :].values inertia = []
for n in range(1 , 11):
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001,
                        random_state= 111 , algorithm='elkan') )
    algorithm.fit(X3) inertia.append(algorithm.inertia_)
```

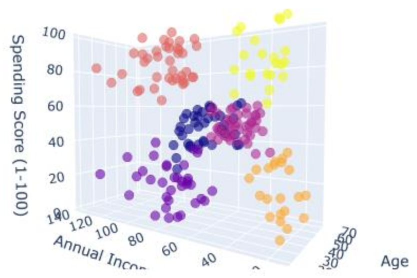
In [25]:
```
plt.figure(1 , figsize = (10 ,6)) plt.plot(np.arange(1 , 11) ,
inertia , 'o')
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5) plt.xlabel('Number of
Clusters') , plt.ylabel('Inertia') plt.show()
```



In [26]:
```
algorithm = (KMeans(n_clusters = 6 ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001,
                        random_state= 111 , algorithm='elkan') )
algorithm.fit(X3)
labels3 = algorithm.labels_
centroids3 = algorithm.cluster_centers_
```

In [27]:
```
df['label3'] = labels3 trace1 =
go.Scatter3d(
    x= df['Age'],
    y= df['Annual Income (k$)'],
    z= df['Spending Score (1-100)'], mode='markers',
     marker=dict(
        color = df['label3'], size= 5,
        line=dict(
            color= df['label3'],
            width= 100
        ),
        opacity=0.6
     )
)
data = [trace1] layout
= go.Layout( #
      margin=dict( #
          l=0,
#             r=0,
#             b=0,
#             t=0
#         )
    title= 'Clusters', scene =
    dict(
            xaxis = dict(title = 'Age'),
            yaxis = dict(title = 'Annual Income'),
            zaxis = dict(title = 'Spending Score (1-100)')
        )
)
fig = go.Figure(data=data, layout=layout)
py.offline.iplot(fig)
```

Clusters



From the above visual representation graph, we can identify some clusters of clients, from a raw behavioural perspective.

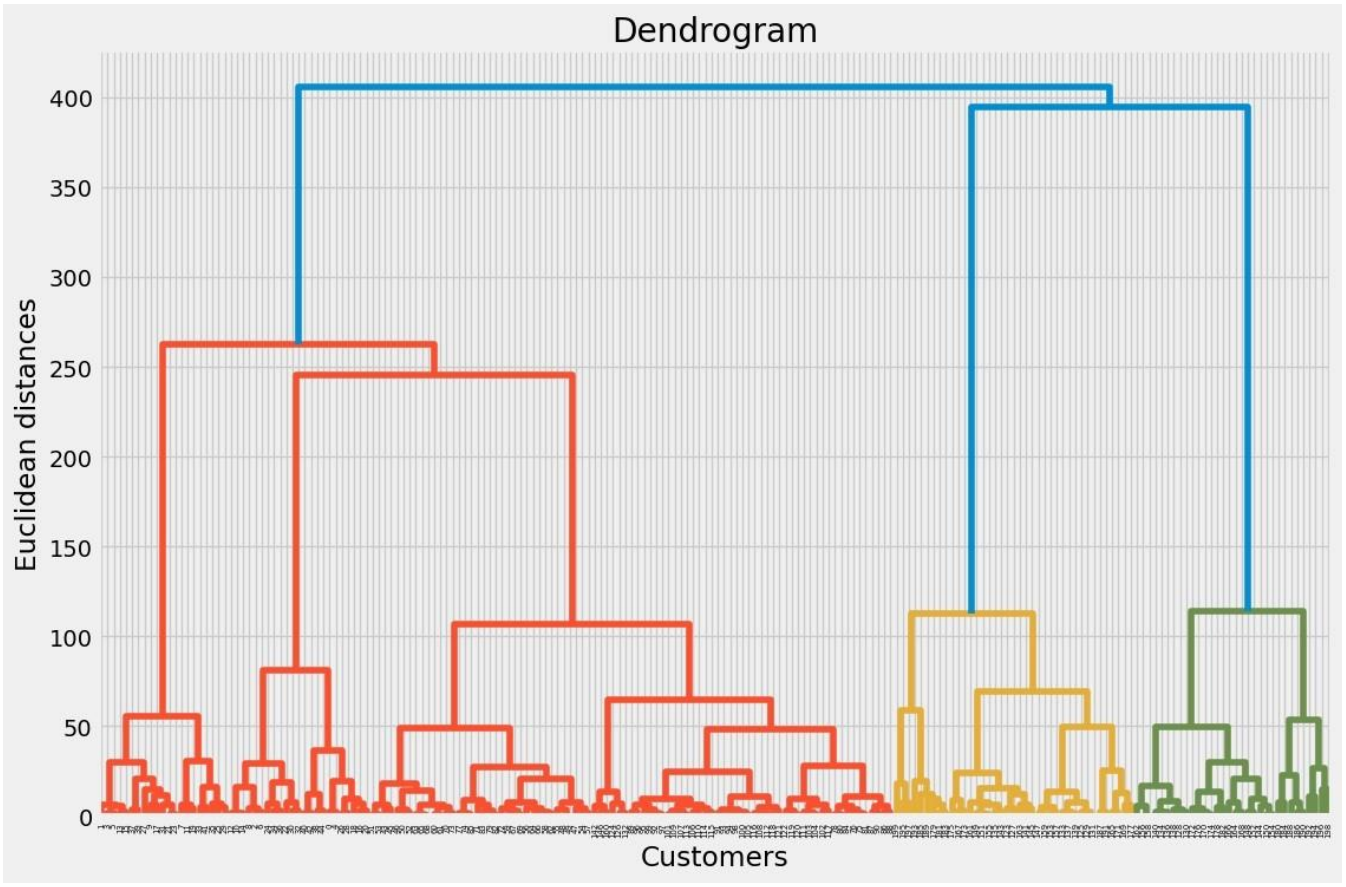
- Cluster 0 (Dark Blue): Young clients (< 40 years old) with low average annual income and average spending score.
- Cluster 1 (Purple): Clients of all ages with high average annual income and average spending score.
- Cluster 2 (Magenta): Clients with high age(> 45 years old), with a an average annual income and average spending score.
- Cluster 3 (Red): Young clients (< 40 years old) with an high annual income and high spending score.
- Cluster 4 (Orange): Clients of all ages, with low annual income and low spending score.
- Cluster 5 (Yellow): Young clients (< 40 years old) with low annual income and average spending score.

Clustering using Hierarchical Clustering technique

Using the dendrogram to find the optimal number of clusters

In [28]:

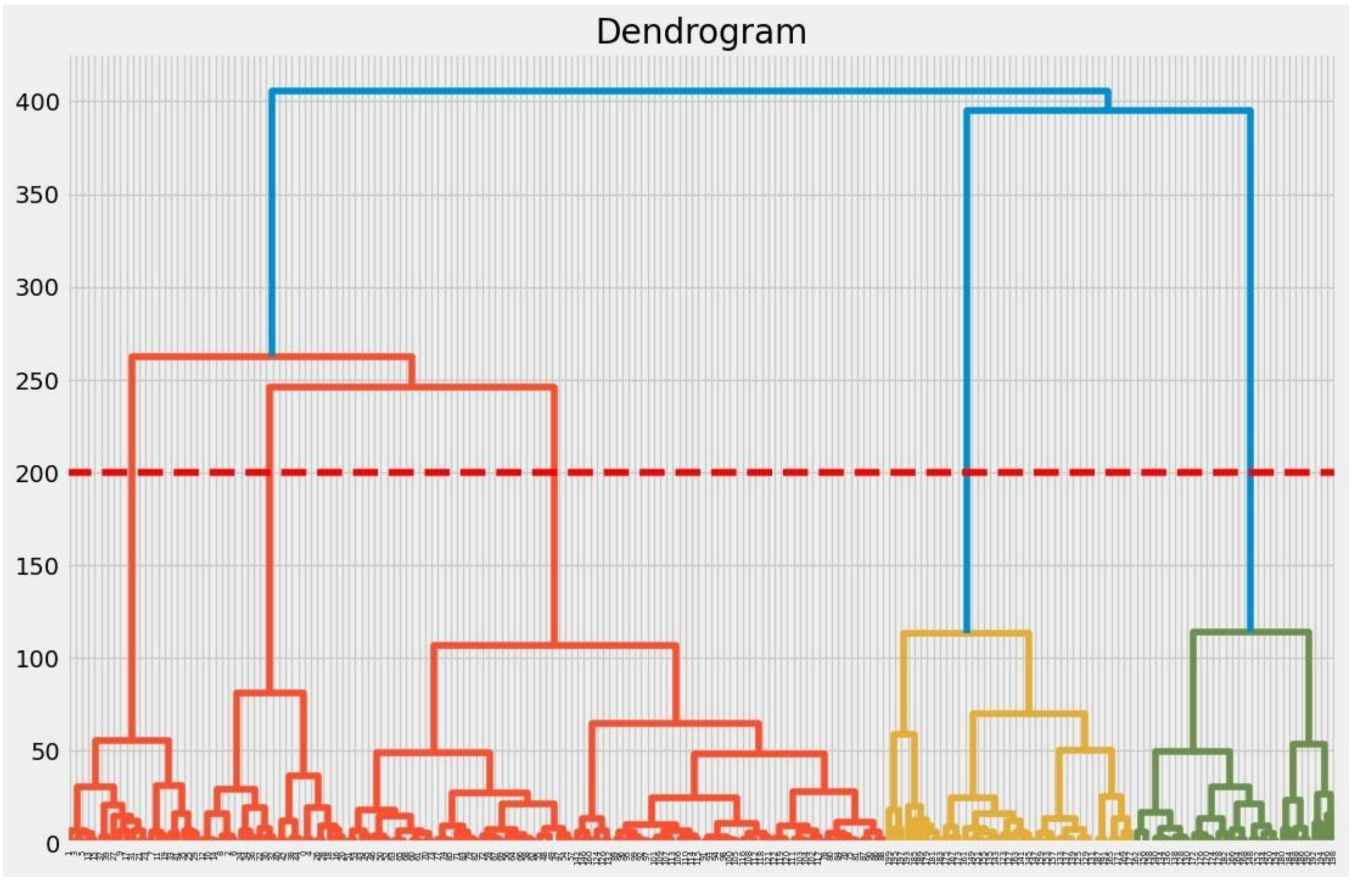
```
import scipy.cluster.hierarchy as sch

X = df.iloc[:, [3, 4]].values plt.figure(1
, figsize = (12 ,8))
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



The x-axis contains the samples and y-axis represents the distance between these samples. The vertical line with maximum distance is the blue line and hence we can decide a threshold of 200 and cut the dendrogram.

```
In [29]: plt.figure(figsize=(12, 8))
plt.title("Dendrogram")
dend = sch.dendrogram(sch.linkage(X, method='ward'))
plt.axhline(y=200, color='r', linestyle='--')
```

```
Out[29]: <matplotlib.lines.Line2D at 0x7fae09516f40>
```



Training the Hierarchical Clustering model on the dataset

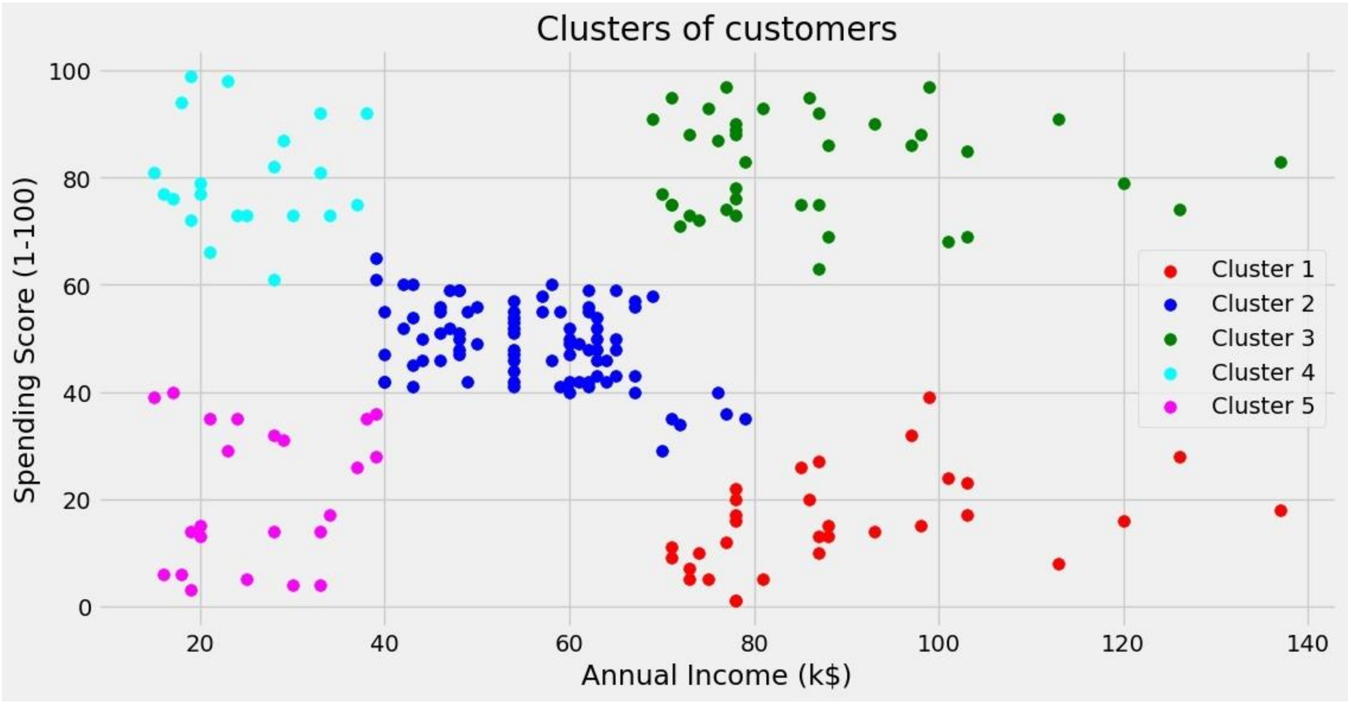
From Dendrogram, we have five clusters as this line cuts the dendrogram at five points. Let’s now apply hierarchical clustering for 5 clusters. Creating an instance of AgglomerativeClustering using the euclidean distance as the measure of distance between points and ward linkage to calculate the proximity of clusters.

```
In [30]: from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward') y_hc =
hc.fit_predict(X)
```

Visualising the clusters

Using shorthand notation to display all the samples belonging to a category as a specific color.

```
In [31]: plt.figure(1 , figsize = (12 ,6))
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 50, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 50, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 50, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 50, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 50, c = 'magenta', label = 'Cluster 5')

plt.title('Clusters of customers') plt.xlabel('Annual
Income (k$)') plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



Yay! We can clearly visualize the five clusters of segmented observations. With the advantage of not having to pre-define the number of clusters gives it quite an edge over k-Means.