Mall_Customer_Segmentation

December 2, 2019

1 Mall Customer Segmentation

1.0.1 1. Motivation

Let's imagine you're owning a supermarket mall and through membership cards, you have some basic data about your customers like Customer ID, age, gender, annual income and spending score, which is something you assign to the customer based on your defined parameters like customer behavior and purchasing data.

The main aim of this problem is learning the purpose of the customer segmentation concepts, also known as market basket analysis, trying to understand customers and sepparate them in different groups according to their preferences, and once the division is done, this information can be given to marketing team so they can plan the strategy accordingly.

All information and data related to this problem can be found here: https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python

1.0.2 2. Data information

This dataset is composed by the following five features:
CustomerID: Unique ID assigned to the customer
Gender: Gender of the customer
Age: Age of the customer
Annual Income (k$): Annual Income of the customer
Spending Score (1-100): Score assigned by the mall based on customer behavior and spending nature.

1.0.3 3. Dependences

Here we can find the libraries we will use in order to develop a solution for this problem.

```
In [1]: import numpy as np
        import pandas as pd
        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")
```

1.0.4 4. Data Exploration

In this section we are doing a little bit of data exploration, checking for null values, object data types and other things we might consider in order to keep our data clean and well structured.

```
In [2]: #We read the csv and print the first 5 rows
        df = pd.read_csv("Mall_Customers.csv")
        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]: #Checking the size of our data
        df.shape
```

```
Out[3]: (200, 5)
```

As we have observed the name of some columns are quite complex and can be changed to simpler names so we can access our data more easily.

```
In [4]: #Changing the name of some columns
        df = df.rename(columns={'Annual Income (k$)': 'Annual_income', 'Spending Score (1-100)
```

```
In [5]: #Looking for null values
        df.isna().sum()
```

```
Out[5]: CustomerID       0
        Gender           0
        Age              0
        Annual_income    0
        Spending_score   0
        dtype: int64
```

```
In [6]: #Checking datatypes
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
CustomerID        200 non-null int64
Gender            200 non-null object
Age               200 non-null int64
Annual_income     200 non-null int64
```

```
Spending_score    200 non-null int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Since Gender is not a numerical value but an object, we are going to replace these values. Female will be 0 and Male will be 1 from now on.

```
In [7]: #Replacing objects for numerical values
        df['Gender'].replace(['Female','Male'], [0,1],inplace=True)
```

```
In [8]: #Checking values have been replaced properly
        df.Gender
```

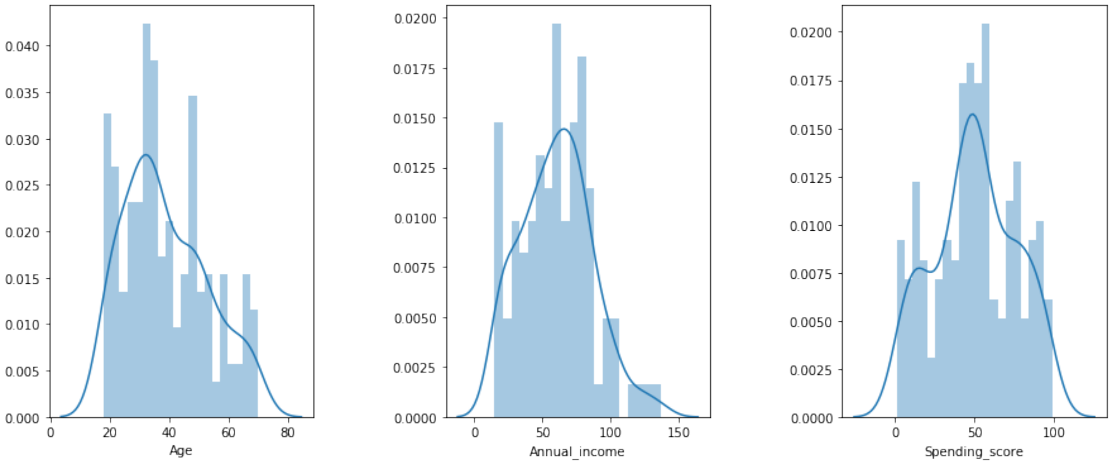
```
Out[8]: 0      1
        1      1
        2      0
        3      0
        4      0
              ..
        195    0
        196    0
        197    1
        198    1
        199    1
        Name: Gender, Length: 200, dtype: int64
```

1.0.5 5. Data Visualization

Now it's the moment to visualize our data and plot important information so we can see the different values our data has and its behaviour. To do so, we are only going to consider the following features: Annual_income, Spending_score and Age. Gender will only be used to make data sepparation so we can differentiate values for men and women.

To begin with, we are plotting the histograms for each of the three features we said we would look into:

```
In [9]: #Density estimation of values using distplot
        plt.figure(1 , figsize = (15 , 6))
        feature_list = ['Age','Annual_income', "Spending_score"]
        feature_listt = ['Age','Annual_income', "Spending_score"]
        pos = 1
        for i in feature_list:
            plt.subplot(1 , 3 , pos)
            plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
            sns.distplot(df[i], bins=20, kde = True)
            pos = pos + 1
        plt.show()
```



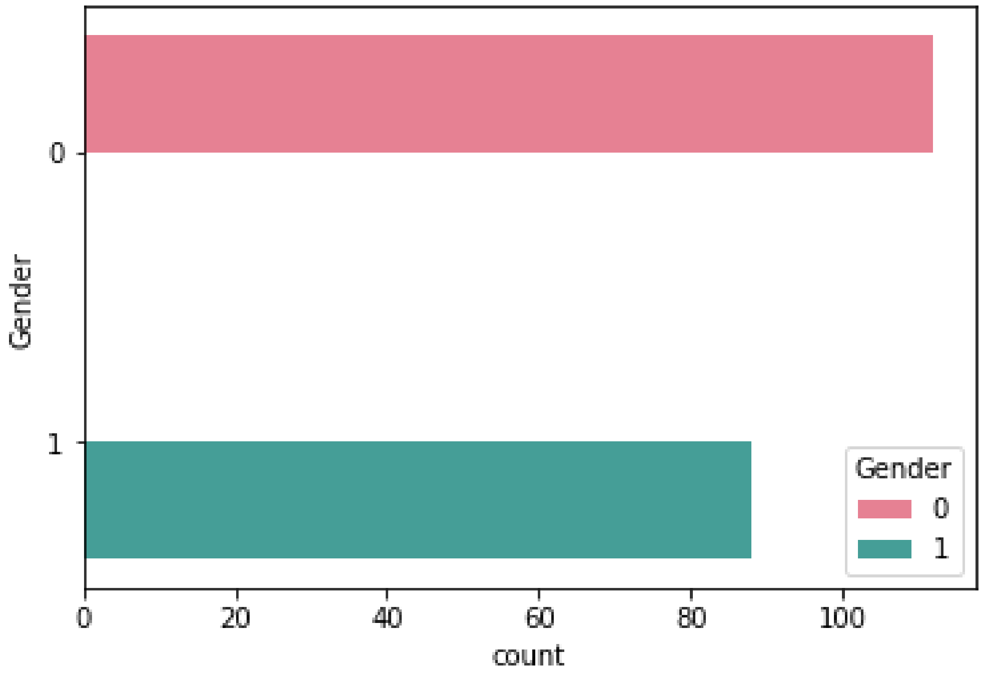
In these histograms we can observe that the distribution of these values resembles a Gaussian distribution, where the vast majority of the values lay in the middle with some exceptions at the extremes.

Now that we have plot the distribution of values through histograms, let's plot the relation between variables using gender as a class distinction. In order to do so we are using the function pairplot given by the Seaborn library, we are using some parameters as well so we can visualize the gender class separation better.

That said, before doing that, let's check how many women and men there are in our data!

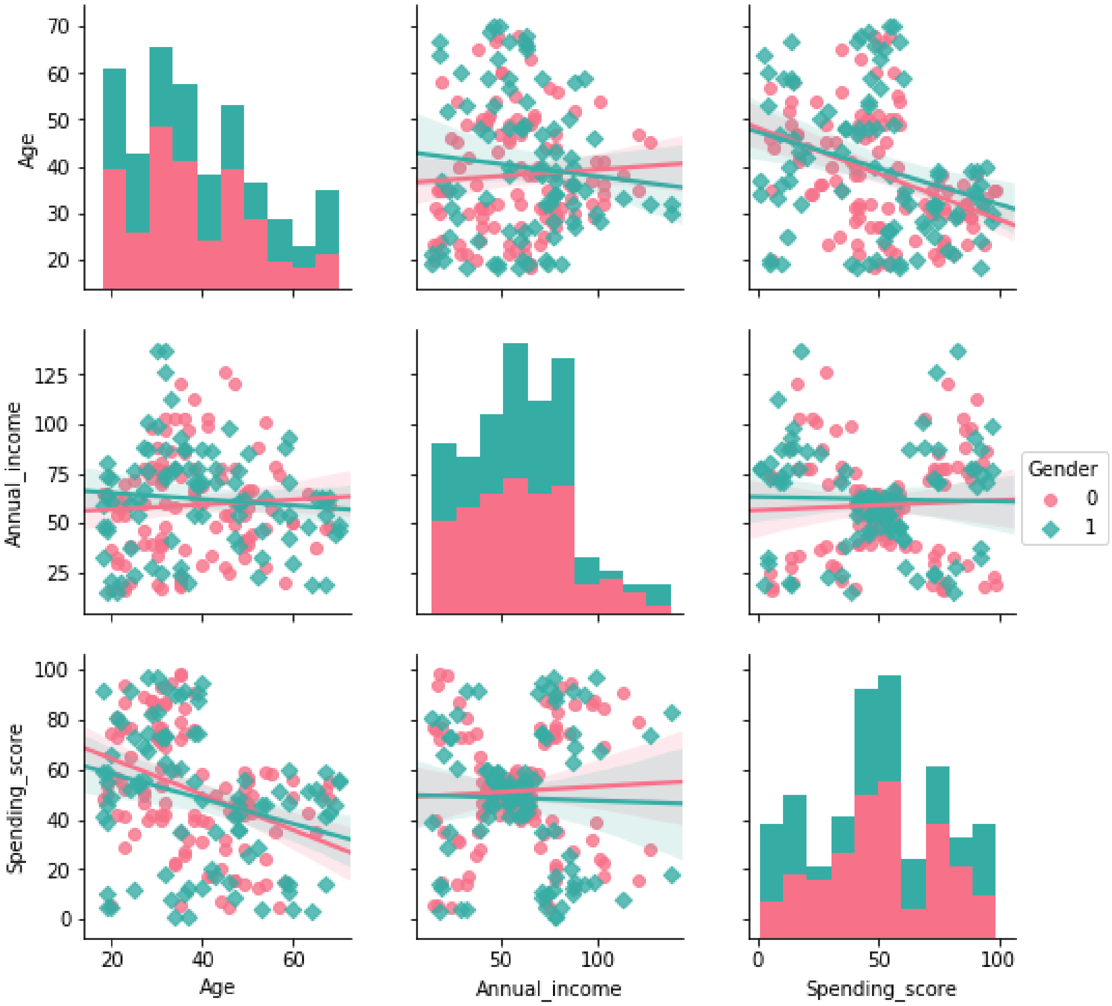
```
In [10]: #Count and plot gender
         sns.countplot(y = 'Gender', data = df, palette="husl", hue = "Gender")
         df["Gender"].value_counts()
```

```
Out[10]: 0    112
         1     88
         Name: Gender, dtype: int64
```



```
In [11]: #Pairplot with variables we want to study
         sns.pairplot(df, vars=["Age", "Annual_income", "Spending_score"],  kind ="reg", hue =

Out[11]: <seaborn.axisgrid.PairGrid at 0x24539a0e128>
```

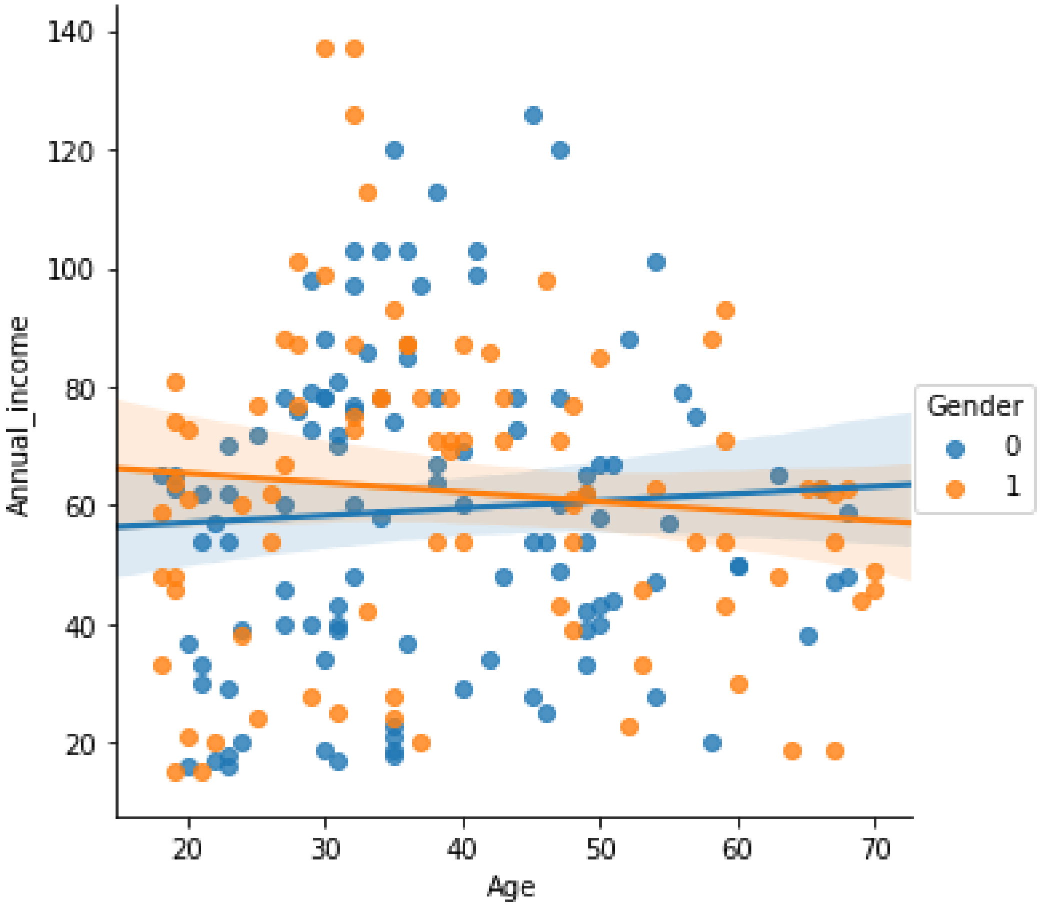


Having plotted these pairplots we can now clearly see the relation between variables. That said let's take a better look at some relations and extract some important information before the clustering process takes place!

Age and Annual Income

```
In [12]: sns.lmplot(x = "Age", y = "Annual_income", data = df, hue = "Gender")
```

```
Out[12]: <seaborn.axisgrid.FacetGrid at 0x2453a4996a0>
```

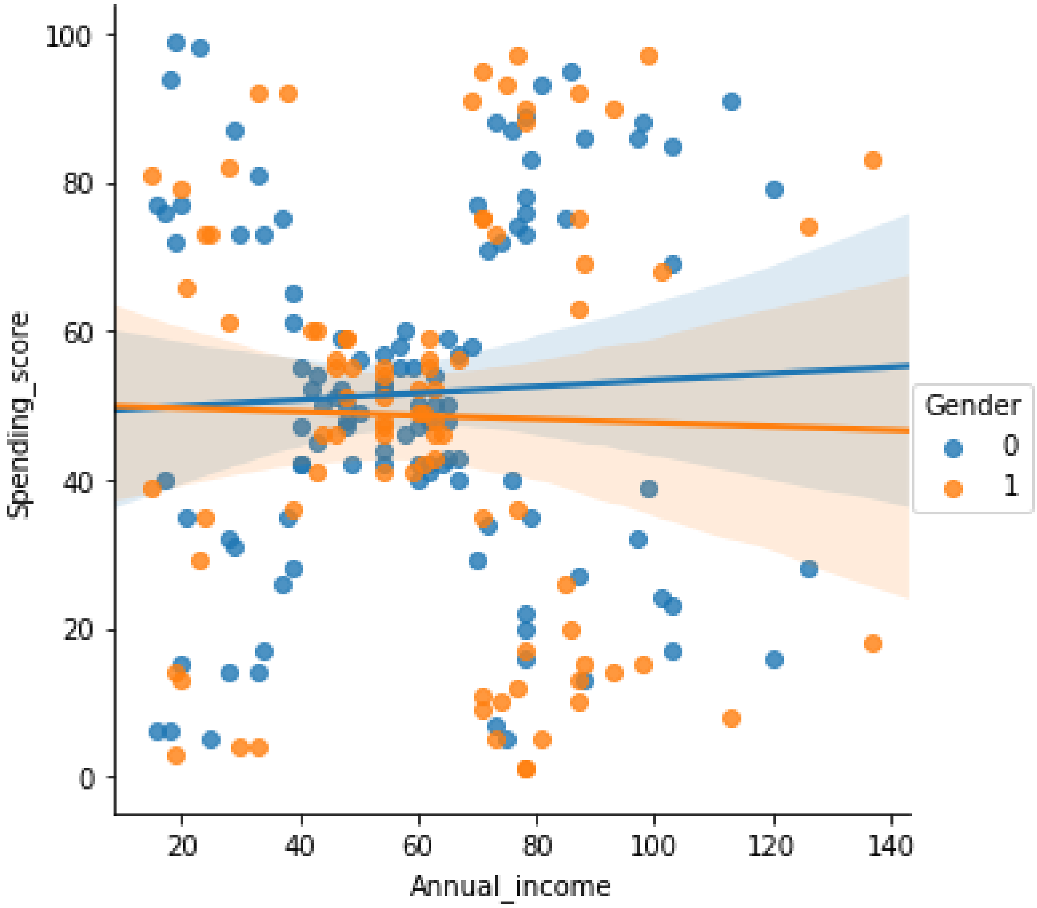


In this graphic we can clearly see how people in their thirties, forties and fifthies tend to earn more money annually than the ones younger than thrity or older than fifty years old. That is to say that people whose age lays between thirty and fifty years old seem to get better jobs since they might be better prepared and are already more experienced than younglings or older people. In the graphic we can also see how males tend to earn a little bit more money than females, at least until fifty years old.

Spending Score and Annual Income

```
In [13]: sns.lmplot(x = "Annual_income", y = "Spending_score", data = df, hue = "Gender")
```

```
Out[13]: <seaborn.axisgrid.FacetGrid at 0x2453a48fa58>
```

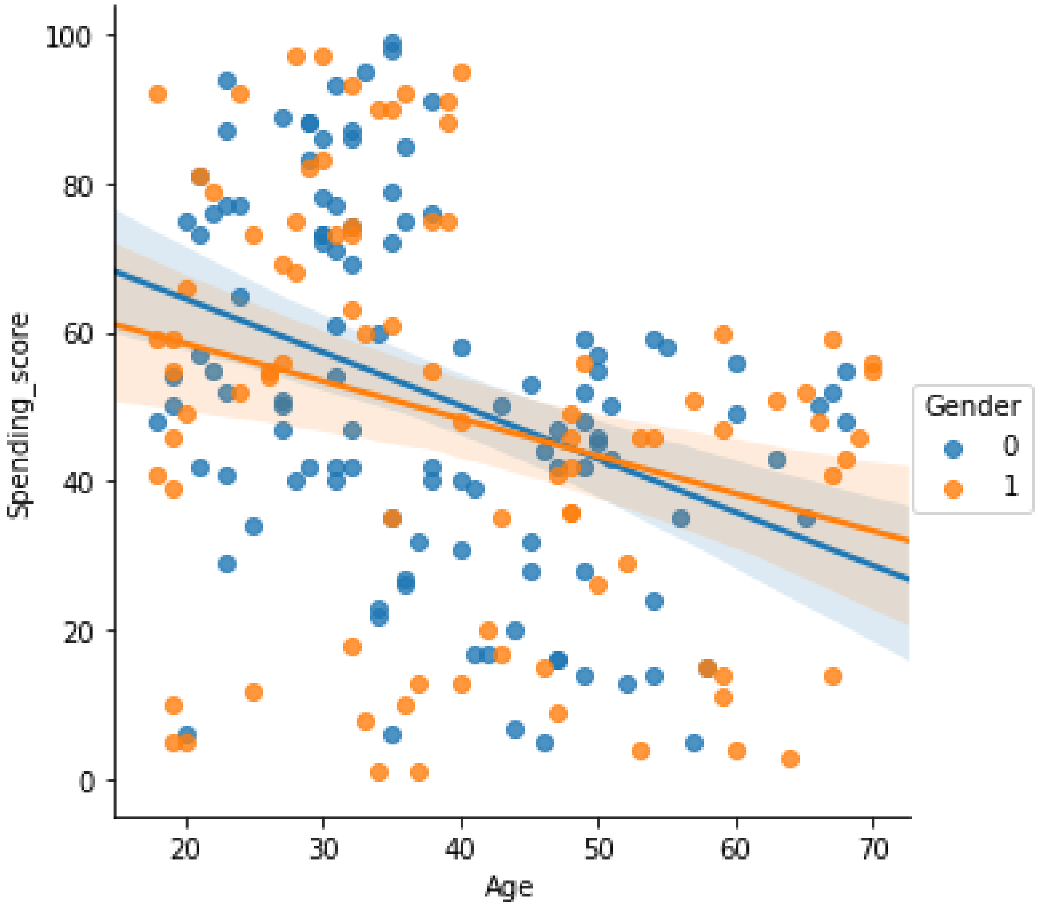


Here we can observe how a better annual income leads to having a higher spending score, specially for women. However the correlation between these two variables isn't that big, we seem to find the majority of people in the middle, people who have decent salaries and have a reasonably high spending score.

Age and Spending Score

```
In [14]: sns.lmplot(x = "Age", y = "Spending_score", data = df, hue = "Gender")
```

```
Out[14]: <seaborn.axisgrid.FacetGrid at 0x2453a6194a8>
```



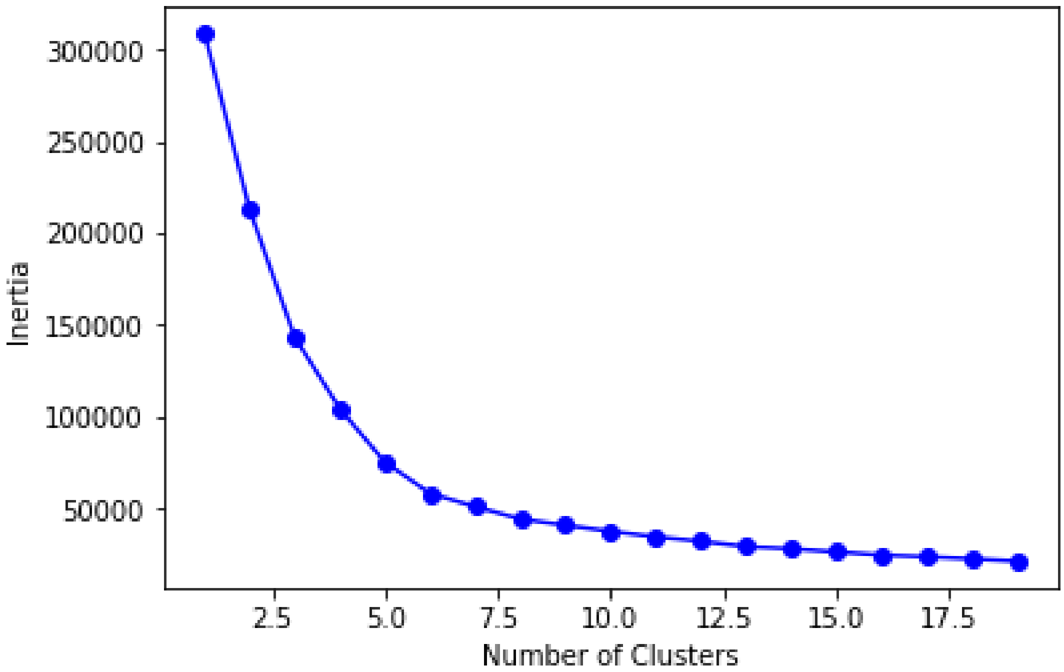
In this last graphic we get to see something we could predict, young people tend to spend way more than older people. That can be due to many reasons: young people usually have more free time than old people, shopping malls tend to have shops that target young people such as videogames and tech stores... and so on.

1.0.6 6. Selecting Number of Clusters

Now that we have already understood this dataset a little bit it's time to decide the amount of clusters we want to divide our data in. To do so, we are going to use the Elbow Method.

```
In [15]: #Creating values for the elbow
         X = df.loc[:,["Age", "Annual_income", "Spending_score"]]
         inertia = []
         k = range(1,20)
         for i in k:
             means_k = KMeans(n_clusters=i, random_state=0)
             means_k.fit(X)
             inertia.append(means_k.inertia_)
```

```
In [16]: #Plotting the elbow
         plt.plot(k , inertia , 'bo-')
         plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
         plt.show()
```



The elbow method is used to determine the optimal number of clusters in k-means clustering. The elbow method plots the value of the cost function produced by different values of k and one should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. In this particular example, if we imagine the line in the graphic is an arm, the elbow can be found, approximately, where the number of clusters is equal to 5. Therefore we are selecting **5** as the number of clusters to divide our data in.

1.0.7 7. Clustering

In the process of clustering we will not be considering the gender factor anymore. The first main reason of why we do take this approach is because the difference between male and female in this data is not particularly high and making a gender differentiaton won't provide that much information. The second and not least important reason is the fact that stores, in general, hardly ever target a specific gender anymore, in almost every store in a mall male and female products can be found.

Additionally we do not want to interfere in the process of unsupervised learning, we will leave the algorith do its job and once it's finished we will analyze the results and extract conclusions and knowledge.

```
In [17]: #Training kmeans with 5 clusters
         means_k = KMeans(n_clusters=5, random_state=0)
```

```
means_k.fit(X)
labels = means_k.labels_
centroids = means_k.cluster_centers_
```

As we can observe, the K-means algorith has already done its work and now it's time to plot the information obtained by it so we can visualize the different clusters and analyze them.

```
In [18]: #Create a 3d plot to view the data sepparation made by Kmeans
         trace1 = go.Scatter3d(
             x= X['Spending_score'],
             y= X['Annual_income'],
             z= X['Age'],
             mode='markers',
              marker=dict(
                 color = labels,
                 size= 10,
                 line=dict(
                     color= labels,
                 ),
                 opacity = 0.9
              )
         )
         layout = go.Layout(
             title= 'Clusters',
             scene = dict(
                     xaxis = dict(title  = 'Spending_score'),
                     yaxis = dict(title  = 'Annual_income'),
                     zaxis = dict(title  = 'Age')
                 )
         )
         fig = go.Figure(data=trace1, layout=layout)
         py.offline.iplot(fig)
```

After plotting the results obtained by K-means in this 3D graphic, it's our job now to identify and describe the five clusters that have been created: - Yellow Cluster - The yellow cluster groups young people with moderate to low annual income who actually spend a lot. - Purple Cluster - The purple cluster groups reasonably young people with pretty decent salaries who spend a lot. - Pink Cluster - The pink cluster basically groups people of all ages whose salary isn't pretty high and their spending score is moderate. - Orange Cluster - The orange cluster groups people who actually have pretty good salaries and barely spend money, their age usually lays between thirty and sixty years. - Blue Cluster - The blue cluster groups whose salary is pretty low and don't spend much money in stores, they are people of all ages.

1.0.8 8. Conclusions

After developing a solution for this problem, we have come to the following conclusions: - KMeans Clustering is a powerful technique in order to achieve a decent customer segmentation. - Customer segmentation is a good way to understand the behaviour of different customers and plan a good marketing strategy accordingly. - There isn't much difference between the spending score of women and men, which leads us to think that our behaviour when it comes to shopping is pretty similar. - Observing the clustering graphic, it can be clearly observed that the ones who spend more money in malls are young people. That is to say they are the main target when it comes to marketing, so doing deeper studies about what they are interested in may lead to higher profits. - Althought younglings seem to be the ones spending the most, we can’t forget there are more people we have to consider, like people who belong to the pink cluster, they are what we would commonly name after "middle class" and they seem to be the biggest cluster. - Promoting discounts on some shops can be something of interest to those who don’t actually spend a lot and they may end up spending more!