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.

The most common ways in which businesses segment their customer base are:

1. **Demographic information**, such as gender, age, familial and marital status, income, education, and occupation.
2. **Geographical information**, which differs depending on the scope of the company. For localized businesses, this info might pertain to specific towns or counties. For larger companies, it might mean a customer's city, state, or even country of residence.
3. **Psychographics**, such as social class, lifestyle, and personality traits.
4. **Behavioral data**, such as spending and consumption habits, product/service usage, and desired benefits.

Advantages of Customer Segmentation

1. Determine appropriate product pricing.
2. Develop customized marketing campaigns.
3. Design an optimal distribution strategy.
4. Choose specific product features for deployment.

5. Prioritize new product development efforts.

The Challenge

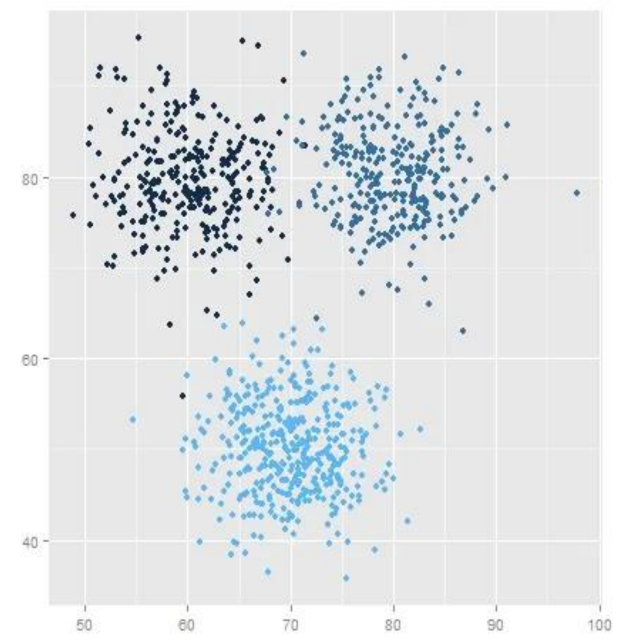
You are owing a supermarket mall and through membership cards, you have some basic data about your customers like Customer ID, age, gender, annual income and spending score. You want to understand the customers like who are the target customers so that the sense can be given to marketing team and plan the strategy accordingly.

K Means Clustering Algorithm

1. Specify number of clusters *K*.

2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.

3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.



Data

This project is a part of the Customer Segmentation Data competition held on Kaggle.

The dataset can be downloaded from the kaggle website which can be found here.

Environment and tools

1. scikit-learn
2. seaborn
3. numpy
4. pandas
5. matplotlib

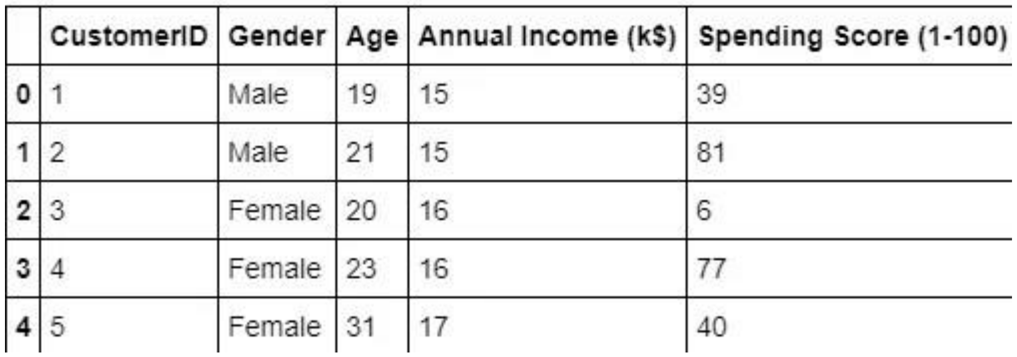
Where is the code?

Without much ado, let’s get started with the code. The complete project on github can be found here.

I started with loading all the libraries and dependencies. The columns in the dataset are customer id, gender, age, income and spending score.

```
import
numpy
as np

         import pandas as pd

         import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("../input/customer-segmentation-tutorial-in-
python/Mall_Customers.csv")

df.head()
```

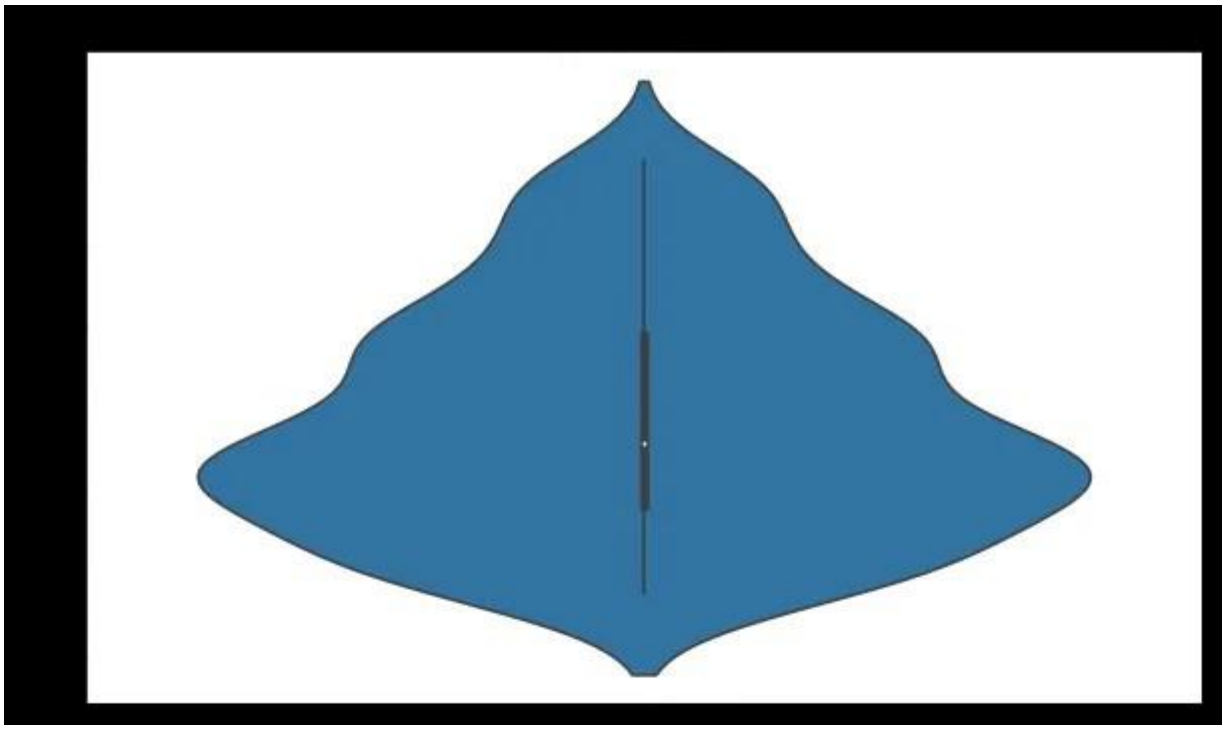


	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

I dropped the id column as that does not seem relevant to the context. Also I plotted the age frequency of customers.

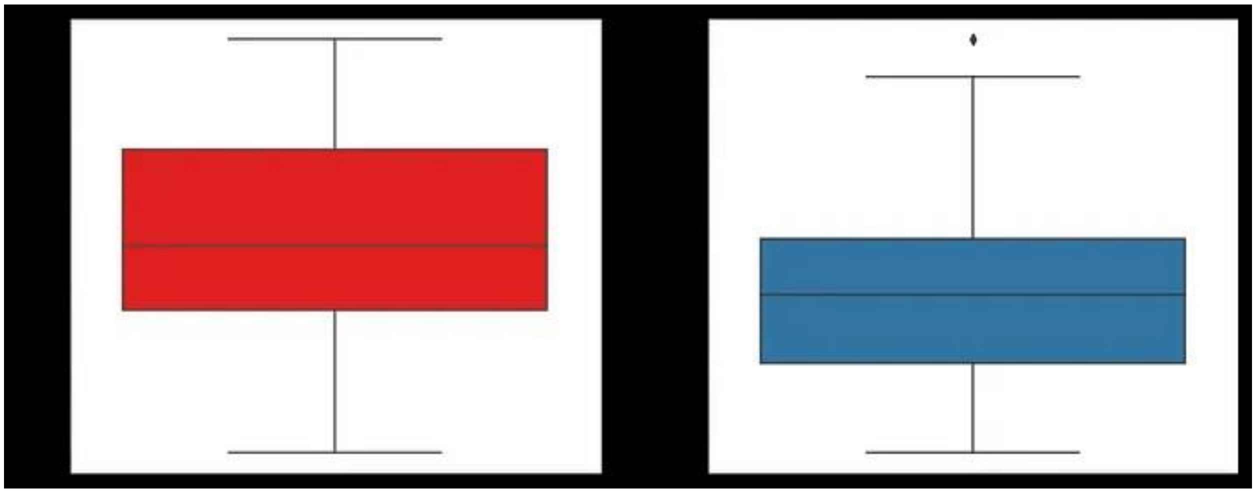
```
df.drop(["CustomerID"],
axis = 1, inplace=True)
```

```
plt.figure(figsize=(10,6))

plt.title("Ages Frequency")

sns.axes_style("dark")

sns.violinplot(y=df["Age"])

plt.show()
```



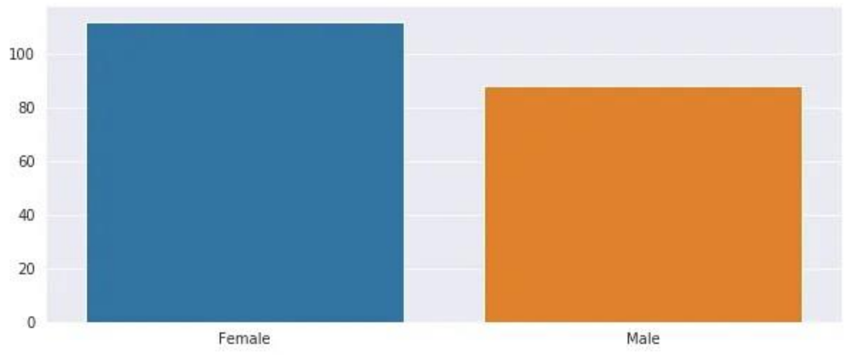
Next I made a box plot of spending score and annual income to better visualize the distribution range. The range of spending score is clearly more than the annual income range.

```
plt.figure(figsize=(15,6))

                                  plt.subplot(1,2,1)

                                  sns.boxplot(y=df["Spending Score (1-100)"], color="red")

                                  plt.subplot(1,2,2)

                                  sns.boxplot(y=df["Annual Income (k$)"])

                                  plt.show ()
```



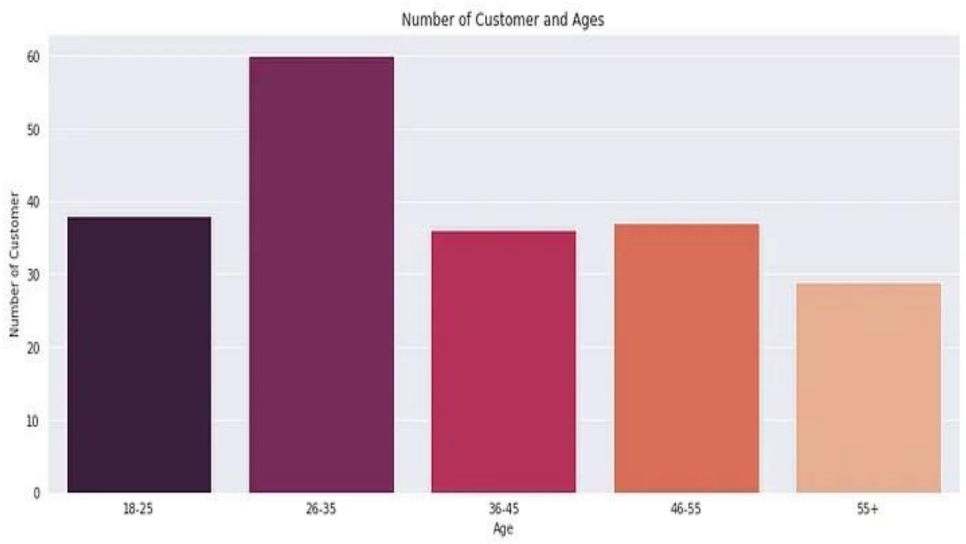
I made a bar plot to check the distribution of male and female population in the dataset. The female population clearly outweighs the male counterpart.

```
genders =
df.Gender.value_counts()

                    sns.set_style("darkgrid")

                    plt.figure(figsize=(10,4))

                    sns.barplot(x=genders.index, y=genders.values)

                    plt.show()
```



Next I made a bar plot to check the distribution of number of customers in each age group. Clearly the 26–35 age group outweighs every other age group

```
age18_25 = df.Age[(df.Age <= 25) & (df.Age >= 18)]

age26_35 = df.Age[(df.Age <= 35) & (df.Age >= 26)]

age36_45 = df.Age[(df.Age <= 45) & (df.Age >= 36)]

age46_55 = df.Age[(df.Age <= 55) & (df.Age >= 46)]

age55above = df.Age[df.Age >= 56]


x = ["18-25","26-35","36-45","46-55","55+"]

y = [len(age18_25.values),len(age26_35.values),len(age36_45.values),len(age46_55.values),len(age55above.values)]


plt.figure(figsize=(15,6))

sns.barplot(x=x, y=y, palette="rocket")

plt.title("Number of Customer and Ages")

plt.xlabel("Age")

plt.ylabel("Number of Customer")

plt.show()
```

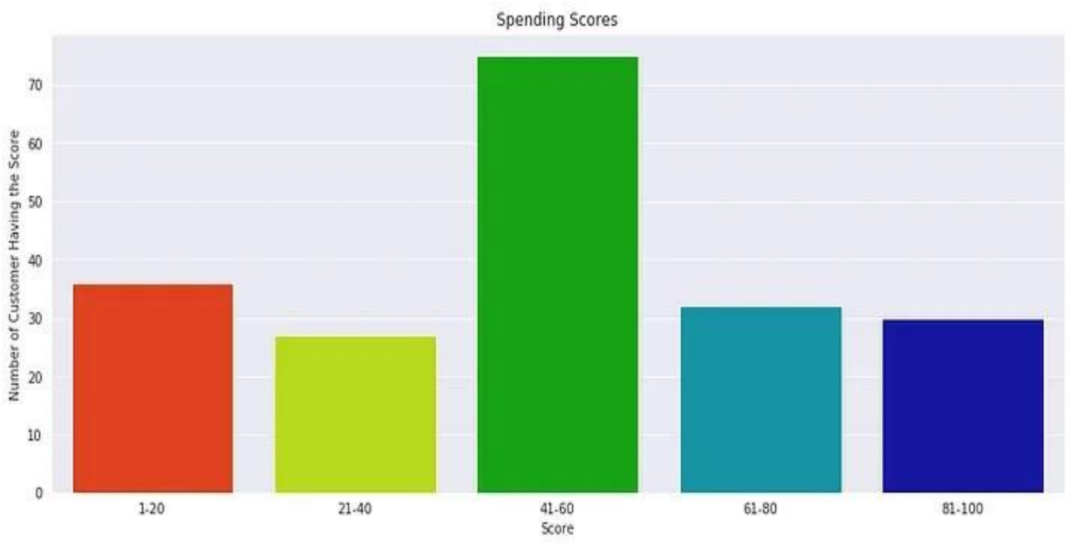


I continued with making a bar plot to visualize the number of customers according to their spending scores. The majority of the customers have spending score in the range 41–60.

```
ss1_20 =
df["Spending Score
(1-
100)"][(df["Spending
Score (1-100)"] >=
1) & (df["Spending
Score (1-100)"] <=
20)]
```

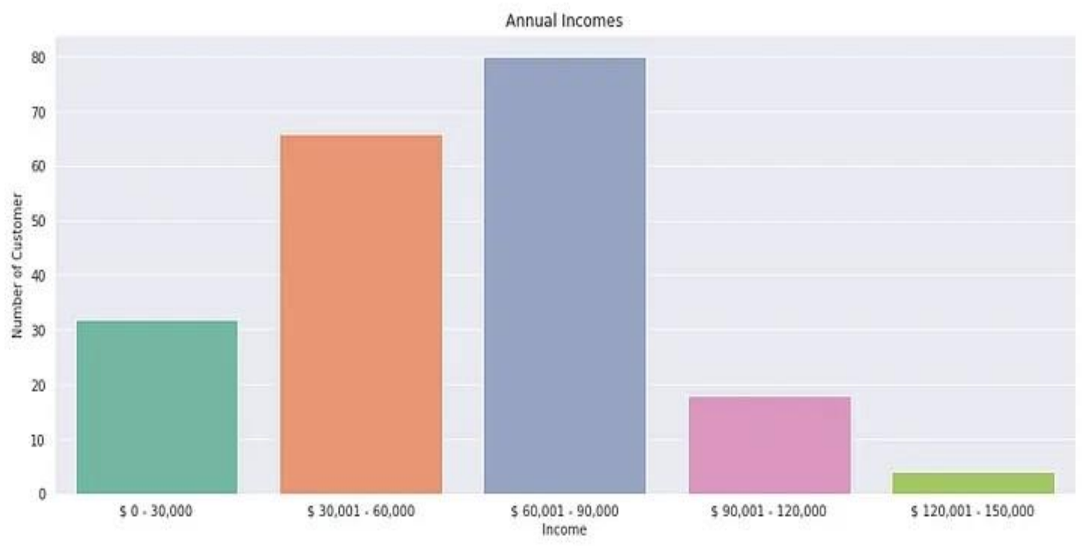
```
ss21_40 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"]
>= 21) & (df["Spending Score (1-100)"] <= 40)]

ss41_60 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"]
>= 41) & (df["Spending Score (1-100)"] <= 60)]

ss61_80 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"]
>= 61) & (df["Spending Score (1-100)"] <= 80)]

ss81_100 = df["Spending Score (1-100)"][(df["Spending Score (1-
100)"] >= 81) & (df["Spending Score (1-100)"] <= 100)]
```

```
ssx = ["1-20", "21-40", "41-60", "61-80", "81-100"]

ssy = [len(ss1_20.values), len(ss21_40.values), len(ss41_60.values),
len(ss61_80.values), len(ss81_100.values)]


plt.figure(figsize=(15,6))

sns.barplot(x=ssx, y=ssy, palette="nipy_spectral_r")

plt.title("Spending Scores")

plt.xlabel("Score")

plt.ylabel("Number of Customer Having the Score")

plt.show()
```



Also I made a bar plot to visualize the number of customers according to their annual income. The majority of the customers have annual income in the range 60000 and 90000.

```
ai0_30 =
df["Annual Income
(k$)"][(df["Annual
Income (k$)"] >=
0) & (df["Annual
Income (k$)"] <=
30)]

ai31_60 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 31) &
(df["Annual Income (k$)"] <= 60)]

ai61_90 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 61) &
(df["Annual Income (k$)"] <= 90)]

ai91_120 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 91) &
(df["Annual Income (k$)"] <= 120)]

ai121_150 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 121)
& (df["Annual Income (k$)"] <= 150)]


aix = ["$ 0 - 30,000", "$ 30,001 - 60,000", "$ 60,001 - 90,000", "$
90,001 - 120,000", "$ 120,001 - 150,000"]

aiy = [len(ai0_30.values), len(ai31_60.values), len(ai61_90.values),
len(ai91_120.values), len(ai121_150.values)]


plt.figure(figsize=(15,6))

sns.barplot(x=aix, y=aiy, palette="Set2")

plt.title("Annual Incomes")

plt.xlabel("Income")

plt.ylabel("Number of Customer")

plt.show()
```



Next I plotted Within Cluster Sum Of Squares (WCSS) against the the number of clusters (K Value) to figure out the optimal number of clusters value. WCSS measures sum of distances of observations from their cluster centroids which is given by the below formula.

$$WCSS = \sum_{i \in n} (X_i - Y_i)^2$$

where *Yi* is centroid for observation *Xi*. The main goal is to maximize number of clusters and in limiting case each data point becomes its own cluster centroid.

```
from
sklearn.cluster
import KMeans

                    wcss = []

                    for k in range(1,11):
```

```
        kmeans = KMeans(n_clusters=k, init="k-means++")

        kmeans.fit(df.iloc[:,1:])

        wcss.append(kmeans.inertia_)

    plt.figure(figsize=(12,6))

    plt.grid()

    plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")

    plt.xlabel("K Value")

    plt.xticks(np.arange(1,11,1))

    plt.ylabel("WCSS")

    plt.show()
```

The Elbow Method

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.

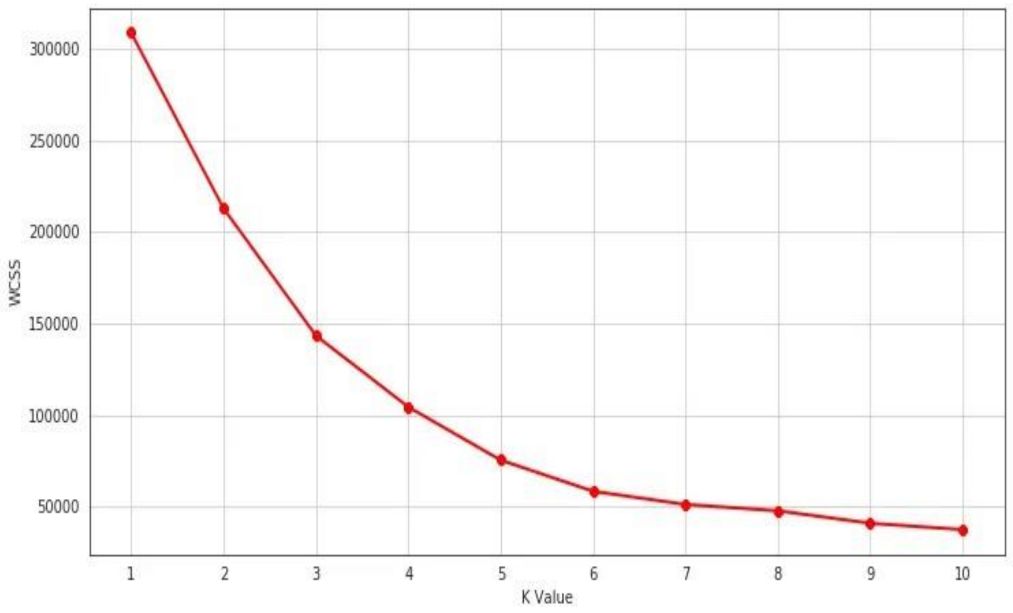
The steps can be summarized in the below steps:

1. Compute K-Means clustering for different values of K by varying K from 1 to 10 clusters.

2. For each K, calculate the total within-cluster sum of square (WCSS).

3. Plot the curve of WCSS vs the number of clusters K.

4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.



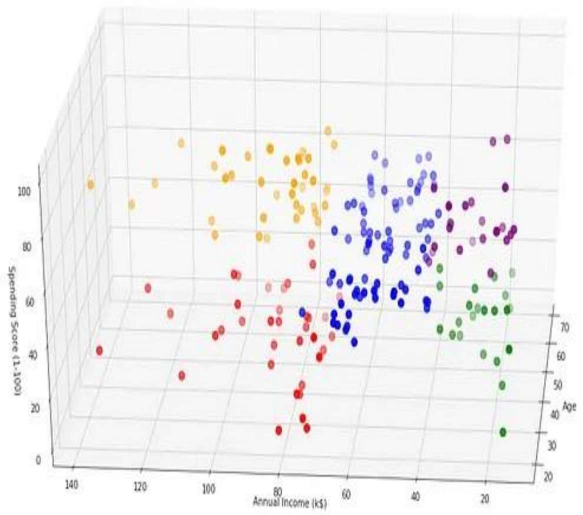
The optimal K value is found to be 5 using the elbow method.

Finally I made a 3D plot to visualize the spending score of the customers with their annual income. The data points are separated into 5 classes which are represented in different colours as shown in the 3D plot.

```
km =
KMeans(n_clusters=5)
                        clusters = km.fit_predict(df.iloc[:,1:])

                        df["label"] = clusters


                        from mpl_toolkits.mplot3d import Axes3D

                        import matplotlib.pyplot as plt

                        import numpy as np

                        import pandas as pd
```

```
fig = plt.figure(figsize=(20,10))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(df.Age[df.label == 0], df["Annual Income (k$)"][df.label
== 0], df["Spending Score (1-100)"][df.label == 0], c='blue', s=60)

ax.scatter(df.Age[df.label == 1], df["Annual Income (k$)"][df.label
== 1], df["Spending Score (1-100)"][df.label == 1], c='red', s=60)

ax.scatter(df.Age[df.label == 2], df["Annual Income (k$)"][df.label
== 2], df["Spending Score (1-100)"][df.label == 2], c='green', s=60)

ax.scatter(df.Age[df.label == 3], df["Annual Income (k$)"][df.label
== 3], df["Spending Score (1-100)"][df.label == 3], c='orange',
s=60)

ax.scatter(df.Age[df.label == 4], df["Annual Income (k$)"][df.label
== 4], df["Spending Score (1-100)"][df.label == 4], c='purple',
s=60)

ax.view_init(30, 185)

plt.xlabel("Age")

plt.ylabel("Annual Income (k$)")

ax.set_zlabel('Spending Score (1-100)')

plt.show()
```

Result



Conclusions

K means clustering is one of the most popular clustering algorithms and usually the first thing practitioners apply when solving clustering tasks to get an idea of the structure of the dataset. The goal of K means is to group data points into distinct non-overlapping subgroups. One of the major application of K means clustering is segmentation of customers to get a better understanding of them which in turn could be used to increase the revenue of the company.