12_Customer_Segmentation_Using_K_Means_Clustering_and_Machine_L

July 9, 2024

0.1 Customer Segmentation Using K-Means Clustering and Machine Learning -Vignesh Prabhu

This project leverages K-Means clustering and machine learning to segment customers based on their behaviors and characteristics. By analyzing patterns within the data, we aim to identify distinct customer groups, providing valuable insights for targeted marketing strategies and improved customer engagement.

Import Dependencies

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

Data Collection and Analysis

[6]:
```
#Loading Data into DataFrame
customer=pd.read_csv("/content/Mall_Customer.csv")
```

[7]:
```
#To print First 5 data's in dataset
customer.head()
```

[7]:

	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

[8]:
```
#To print Last 5 data's in dataset
customer.tail()
```

[8]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74

```
198         199    Male   32                  137                      18
199         200    Male   30                  137                      83
```

[9]:
```
#To check Number Of Row and Columns
customer.shape
```

[9]:
```
(200, 5)
```

[10]:
```
#To Check any Null Values In Dataset
customer.isnull().sum()
```

[10]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

[11]:
```
#Information
customer.info()
```

```
<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]:
```
#Statistical Measures
customer.describe()
```

[12]:

	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

Choosing Annual Income and Spending Columns

[13]:
```
X=customer.iloc[:,[3,4]].values
```

[20]:
```
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]
 [ 23  98]
 [ 24  35]
 [ 24  73]
 [ 25   5]
 [ 25  73]
 [ 28  14]
 [ 28  82]
 [ 28  32]
 [ 28  61]
 [ 29  31]
 [ 29  87]
 [ 30   4]
 [ 30  73]
 [ 33   4]
 [ 33  92]
 [ 33  14]
 [ 33  81]
 [ 34  17]
 [ 34  73]
 [ 37  26]
 [ 37  75]
 [ 38  35]
 [ 38  92]
 [ 39  36]
```

```
[ 39  61]
[ 39  28]
[ 39  65]
[ 40  55]
[ 40  47]
[ 40  42]
[ 40  42]
[ 42  52]
[ 42  60]
[ 43  54]
[ 43  60]
[ 43  45]
[ 43  41]
[ 44  50]
[ 44  46]
[ 46  51]
[ 46  46]
[ 46  56]
[ 46  55]
[ 47  52]
[ 47  59]
[ 48  51]
[ 48  59]
[ 48  50]
[ 48  48]
[ 48  59]
[ 48  47]
[ 49  55]
[ 49  42]
[ 50  49]
[ 50  56]
[ 54  47]
[ 54  54]
[ 54  53]
[ 54  48]
[ 54  52]
[ 54  42]
[ 54  51]
[ 54  55]
[ 54  41]
[ 54  44]
[ 54  57]
[ 54  46]
[ 57  58]
[ 57  55]
[ 58  60]
[ 58  46]
[ 59  55]
```

```
[ 59  41]
[ 60  49]
[ 60  40]
[ 60  42]
[ 60  52]
[ 60  47]
[ 60  50]
[ 61  42]
[ 61  49]
[ 62  41]
[ 62  48]
[ 62  59]
[ 62  55]
[ 62  56]
[ 62  42]
[ 63  50]
[ 63  46]
[ 63  43]
[ 63  48]
[ 63  52]
[ 63  54]
[ 64  42]
[ 64  46]
[ 65  48]
[ 65  50]
[ 65  43]
[ 65  59]
[ 67  43]
[ 67  57]
[ 67  56]
[ 67  40]
[ 69  58]
[ 69  91]
[ 70  29]
[ 70  77]
[ 71  35]
[ 71  95]
[ 71  11]
[ 71  75]
[ 71   9]
[ 71  75]
[ 72  34]
[ 72  71]
[ 73   5]
[ 73  88]
[ 73   7]
[ 73  73]
[ 74  10]
```

```
[ 74  72]
[ 75   5]
[ 75  93]
[ 76  40]
[ 76  87]
[ 77  12]
[ 77  97]
[ 77  36]
[ 77  74]
[ 78  22]
[ 78  90]
[ 78  17]
[ 78  88]
[ 78  20]
[ 78  76]
[ 78  16]
[ 78  89]
[ 78   1]
[ 78  78]
[ 78   1]
[ 78  73]
[ 79  35]
[ 79  83]
[ 81   5]
[ 81  93]
[ 85  26]
[ 85  75]
[ 86  20]
[ 86  95]
[ 87  27]
[ 87  63]
[ 87  13]
[ 87  75]
[ 87  10]
[ 87  92]
[ 88  13]
[ 88  86]
[ 88  15]
[ 88  69]
[ 93  14]
[ 93  90]
[ 97  32]
[ 97  86]
[ 98  15]
[ 98  88]
[ 99  39]
[ 99  97]
[101  24]
```

```
[101  68]
[103  17]
[103  85]
[103  23]
[103  69]
[113   8]
[113  91]
[120  16]
[120  79]
[126  28]
[126  74]
[137  18]
[137  83]]
```

0.2 Choosing Number Of Cluster

WCSS - Within Cluster Sum Of Squares

BCSS - Between Cluster Sum Of Squares

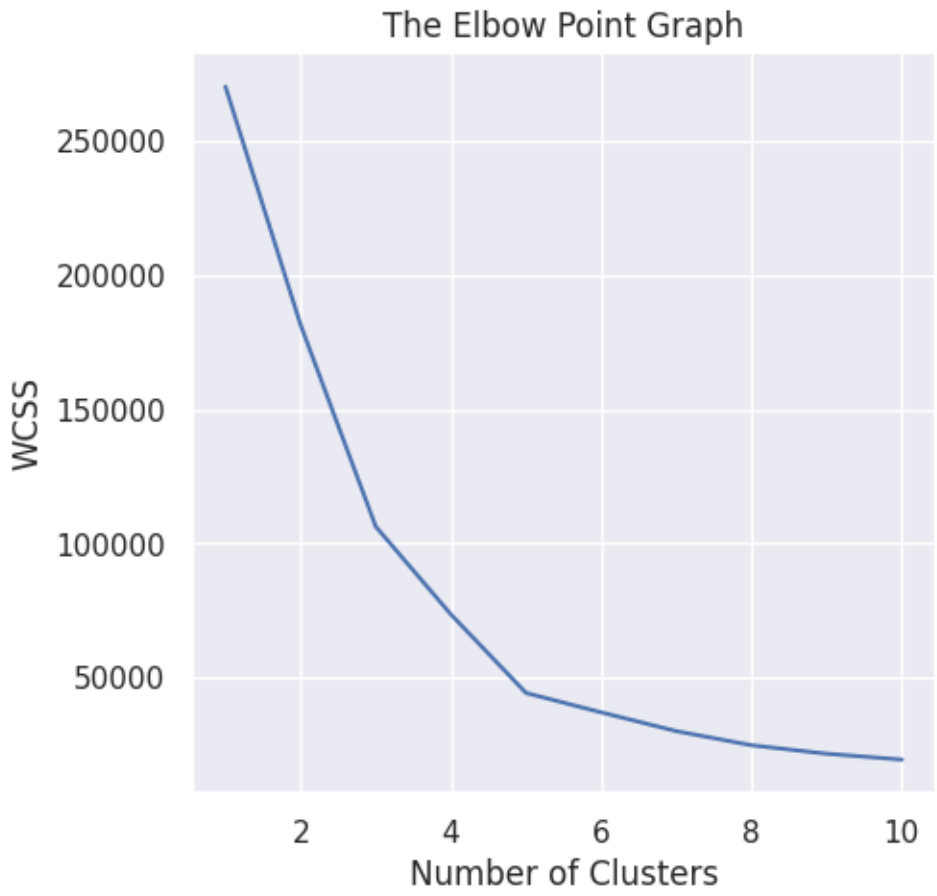
[22]:
```
#Finding WCSS value of different number of clusters
wcss=[] # Empty List

for i in range(1,11):  # range 1 to 10
  kmeans=KMeans(n_clusters=i,init='k-means++',n_init=10,random_state=42)
  kmeans.fit(X)

  wcss.append(kmeans.inertia_)
```

Plot Elbow Graph

[27]:
```
plt.figure(figsize=(5,5))
sns.set()
plt.plot(range(1,11),wcss)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



Optimum Number Of cluster =5

Training The K-means Clustering Model

[31]:
```
kmeans=KMeans(n_clusters=5,init='k-means++',n_init=10,random_state=0)

#Return a label for each data point based on their cluster
y_kmeans=kmeans.fit_predict(X)
print(y_kmeans)
```

```
[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 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 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]
```

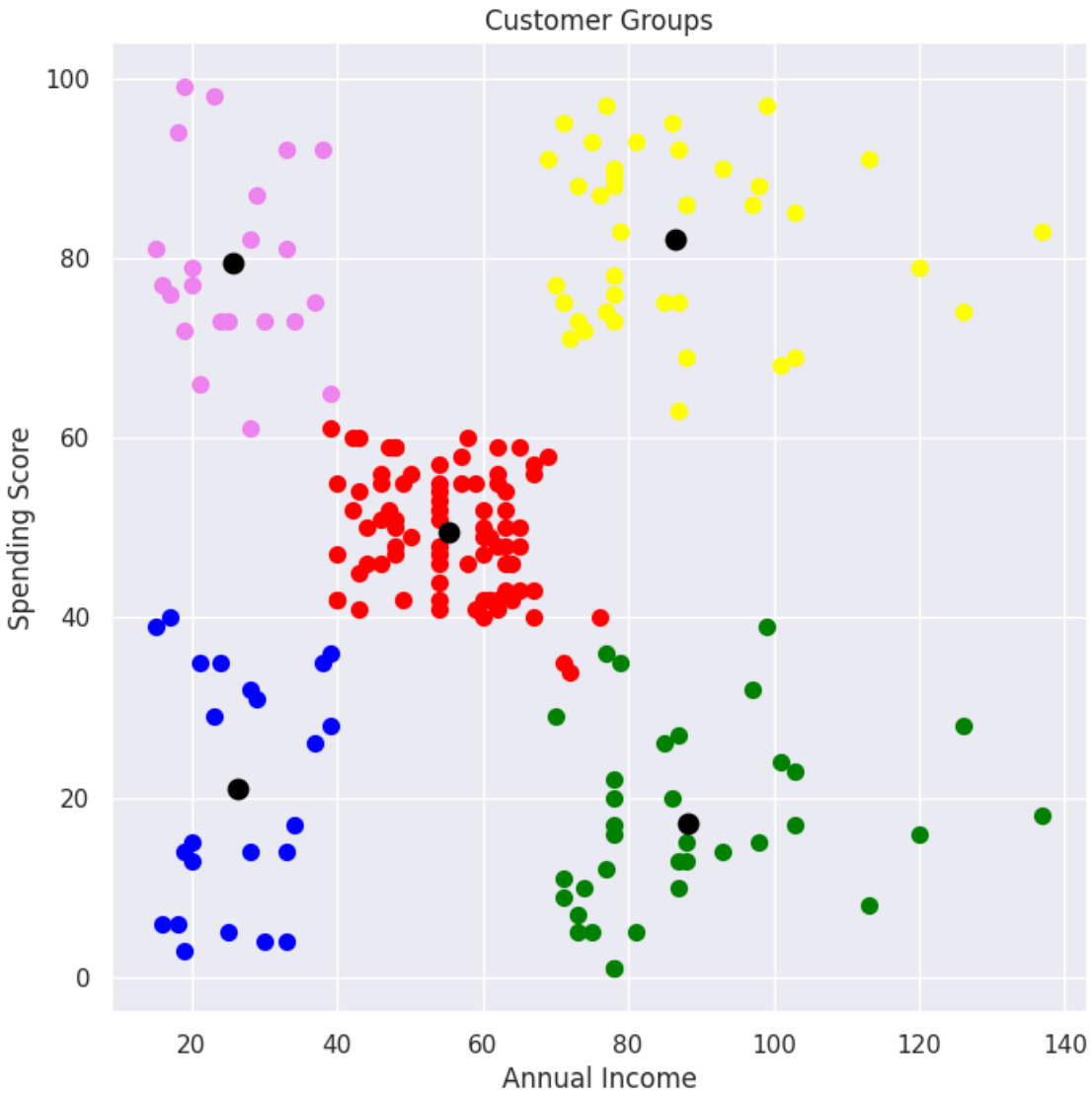
0.3 Visualizing Clusters

[38]:
```
#5 cluster  0,1,2,3,4
plt.figure(figsize=(8,8))
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=50,c='green',label='Cluster1')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=50,c='red',label='Cluster2')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=50,c='yellow',label='Cluster3')
plt.scatter(X[y_kmeans==3,0],X[y_kmeans==3,1],s=50,c='violet',label='Cluster4')
plt.scatter(X[y_kmeans==4,0],X[y_kmeans==4,1],s=50,c='blue',label='Cluster5')


#Plot Centroid
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:
 ↪,1],s=80,c='black',label='Centroid')

plt.title('Customer Groups')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()


#sns.scatterplot(x='Annual Income (k$)',y='Spending Score␣
 ↪(1-100)',data=customer,hue=y_kmeans)
```



K-means clustering has empowered us to unlock valuable customer insights and tailor our marketing strategies with precision. By segmenting our diverse customer base, we've driven higher engagement, increased conversions, and elevated customer satisfaction. This data-driven approach continues to fuel our growth and adaptability in an ever-evolving market.

1 Thank You !

[ ]: