IBM Assignment 4

October 26, 2022

[156]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import math
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

Load dataset

[129]:
```
df = pd.read_csv("/content/drive/MyDrive/Mall_Customers.csv")
df = pd.DataFrame(df)
df.head()
```

[129]:

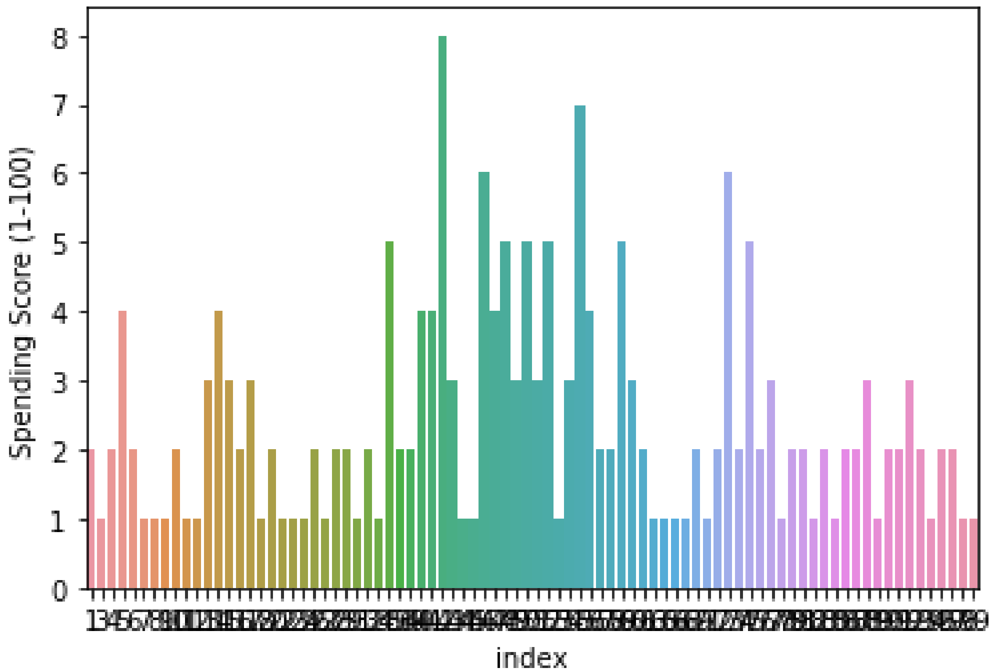
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

[130]:
```
df = df.drop('CustomerID', axis='columns')
```

Univariate analysis

[131]:
```
age = df['Spending Score (1-100)'].value_counts().reset_index()
# barplot
sns.barplot(data=age, x='index', y='Spending Score (1-100)')
```

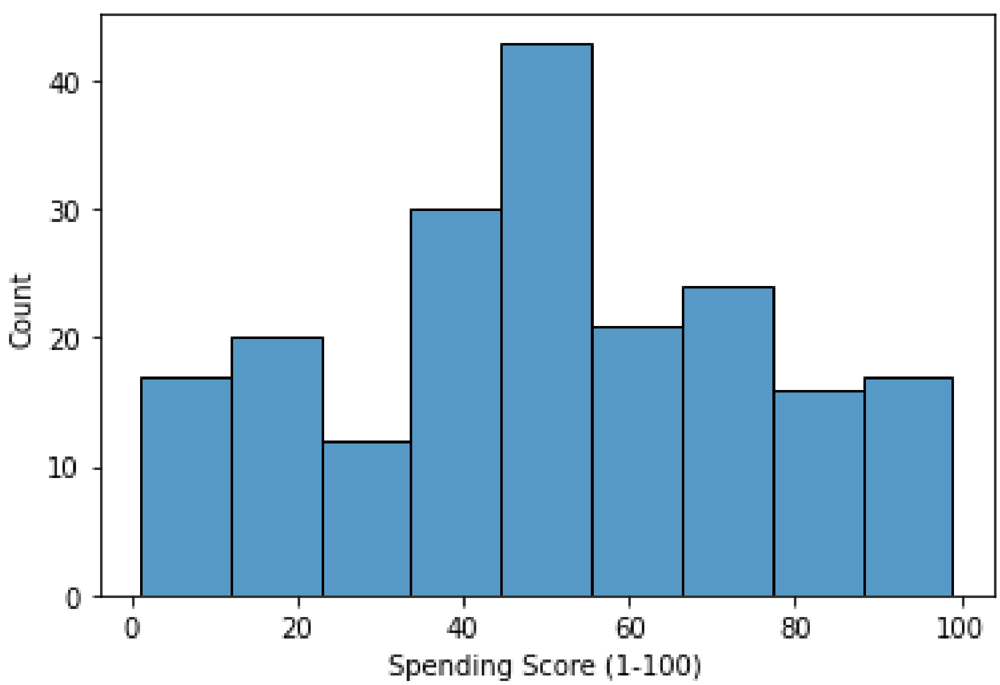
[131]: <matplotlib.axes._subplots.AxesSubplot at 0x7fae6e554b10>



[132]:

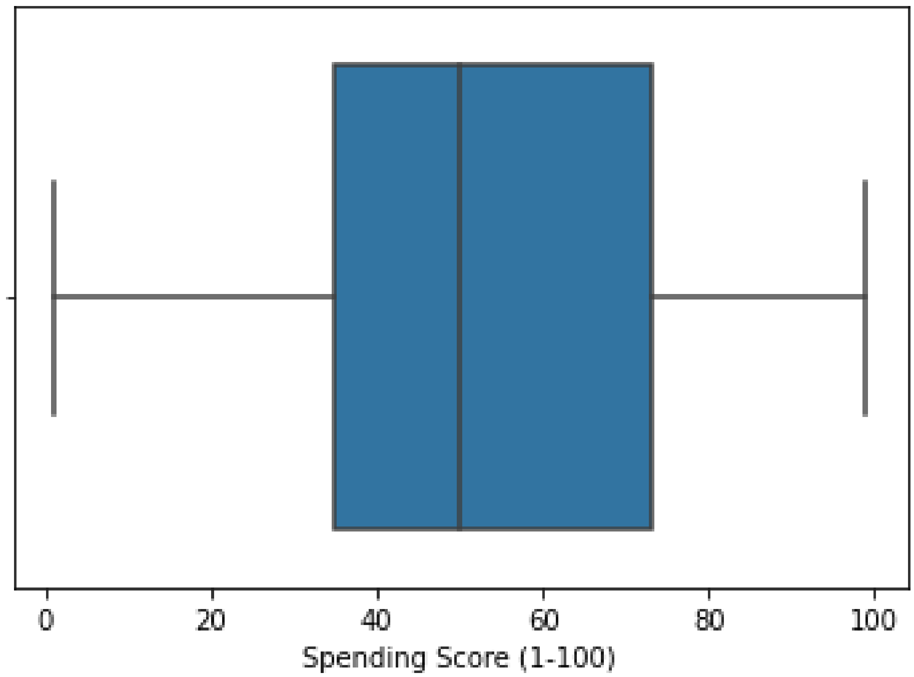
```
#histplot
sns.histplot(x=df['Spending Score (1-100)'])
```

[132]: <matplotlib.axes._subplots.AxesSubplot at 0x7fae6e71fa50>



[133]:
```python
# boxplot
sns.boxplot(x=df['Spending Score (1-100)'])
```

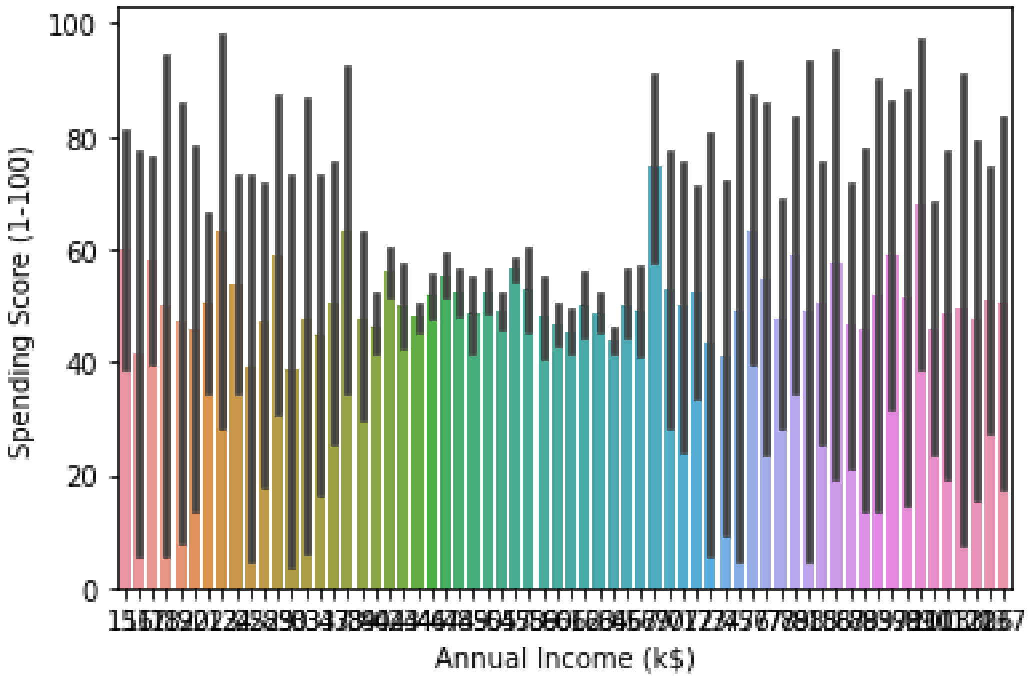
[133]: <matplotlib.axes._subplots.AxesSubplot at 0x7fae6e7438d0>



Bivariate analysis

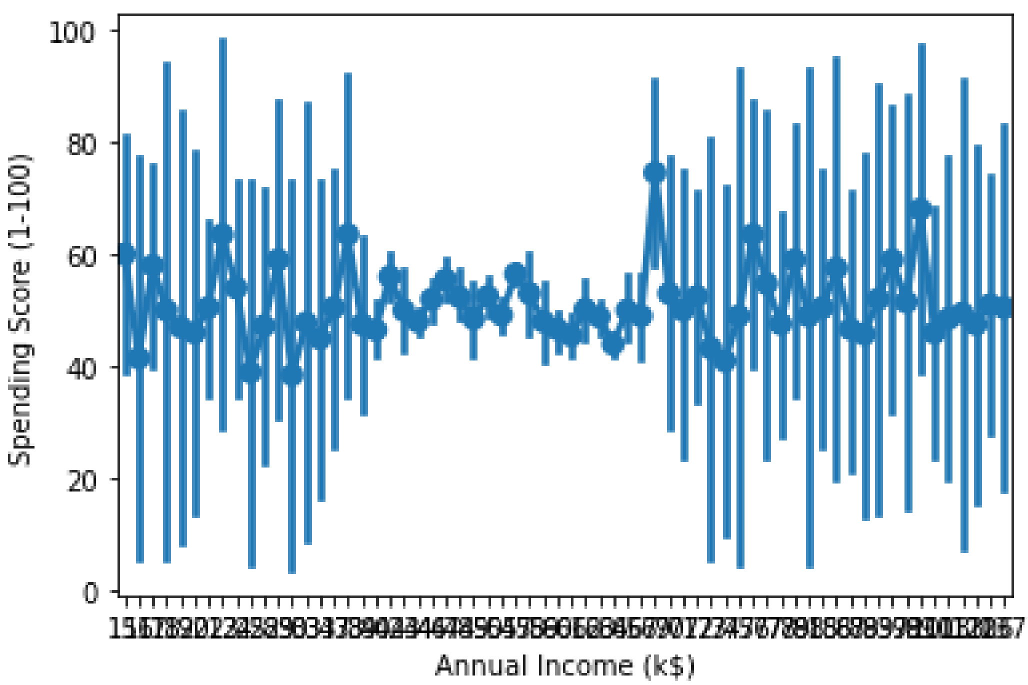
[134]:
```python
#barplot
sns.barplot(x=df['Annual Income (k$)'], y=df['Spending Score (1-100)'])
```

[134]: <matplotlib.axes._subplots.AxesSubplot at 0x7fae6e2e4fd0>



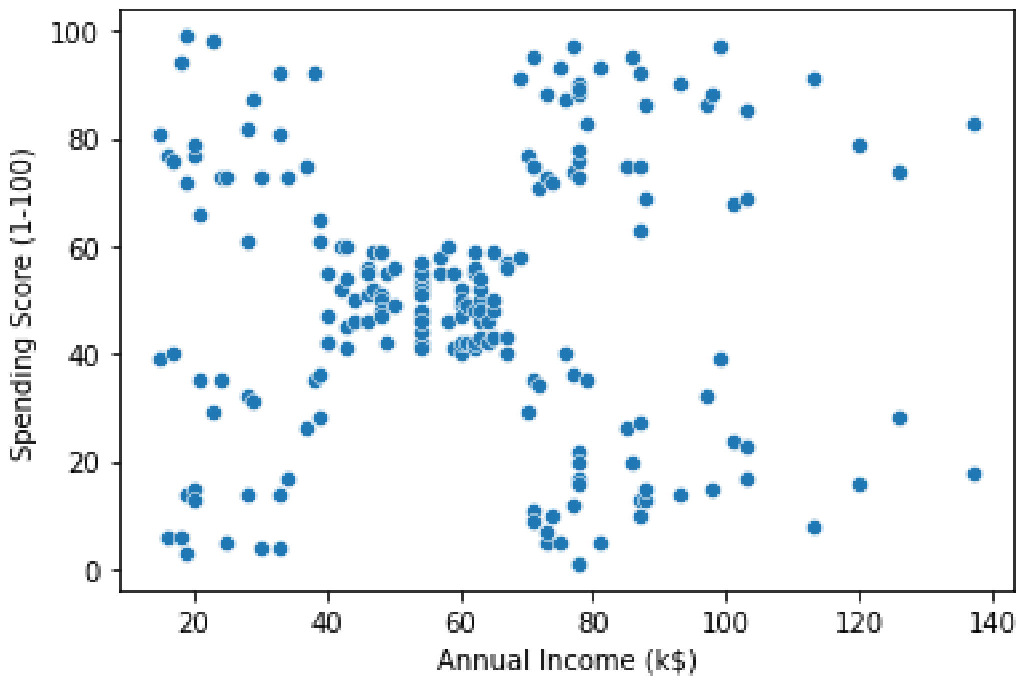
[135]:
```
#pointplot
sns.pointplot(x=df['Annual Income (k$)'], y=df['Spending Score (1-100)'])
```

[135]: <matplotlib.axes._subplots.AxesSubplot at 0x7fae6e325f50>



[136]:
```
#scatter plot
sns.scatterplot(x=df['Annual Income (k$)'], y=df['Spending Score (1-100)'])
```

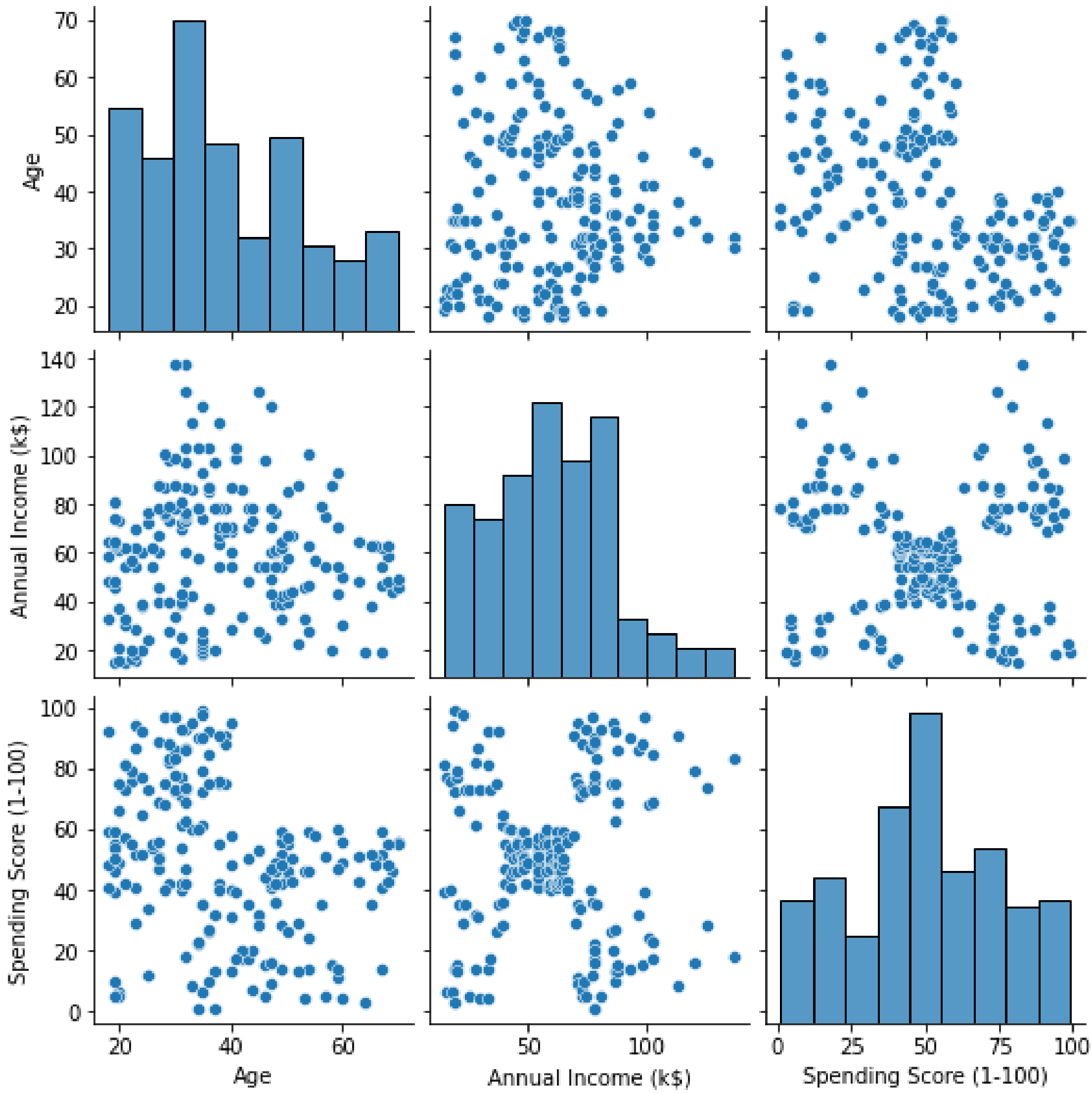
[136]: <matplotlib.axes._subplots.AxesSubplot at 0x7fae6dea9050>



Multivariate analysis

[137]:
```
#pairplot
sns.pairplot(data = df[["Gender",         "Age"         ,"Annual Income (k$)",␣
↪         "Spending Score (1-100)"]])
```

[137]: <seaborn.axisgrid.PairGrid at 0x7fae6e7f5990>



Descriptive statistics

[138]:
```
df.describe()
```

[138]:

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

Missing values and how to deal with them

[139]:
```
df.isnull().sum()
```

[139]:
```
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

[140]:
```
df.isna().sum()
# no missing values
```

[140]:
```
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Find the outliers and replace them outliers

[141]:
```
# replacing numerical outliers with lower and upper limits respectively

for i in df:
  if df[i].dtype=='int64'or df[i].dtypes=='float64':
    q1=df[i].quantile(0.25)
    q3=df[i].quantile(0.75)
    iqr=q3-q1
    upper=q3+1.5*iqr
    lower=q1-1.5*iqr
    df[i]=np.where(df[i] >upper, upper, df[i])
    df[i]=np.where(df[i] <lower, lower, df[i])
```

Check for categorical columns and perform encoding

[142]:
```
# identified and encoded the categorical values
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
for i in df:
  if df[i].dtype=='object' or df[i].dtype=='category':
    print(i)
    df[i]=encoder.fit_transform(df[i])
df.head()
```

```
Gender
```

[142]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19.0	15.0	39.0

```
1        1  21.0                 15.0                     81.0
2        0  20.0                 16.0                      6.0
3        0  23.0                 16.0                     77.0
4        0  31.0                 17.0                     40.0
```

Split the data into dependent and independent variables.

[143]:
```
# independent variables
X = df.iloc[:, :-1].values
```

[144]:
```
# dependent variables
Y = df.iloc[:, -1].values
```

Scale independent variables

[146]:
```
x = scale(df[["Gender",         "Age"         ,"Annual Income (k$)"]])
```

Split the data into training and testing

[147]:
```
X = df.iloc[:, 0:3]
X
```

[147]:
```
     Gender   Age  Annual Income (k$)
0         1  19.0               15.00
1         1  21.0               15.00
2         0  20.0               16.00
3         0  23.0               16.00
4         0  31.0               17.00
..      …     …                 …
195       0  35.0              120.00
196       0  45.0              126.00
197       1  32.0              126.00
198       1  32.0              132.75
199       1  30.0              132.75

[200 rows x 3 columns]
```

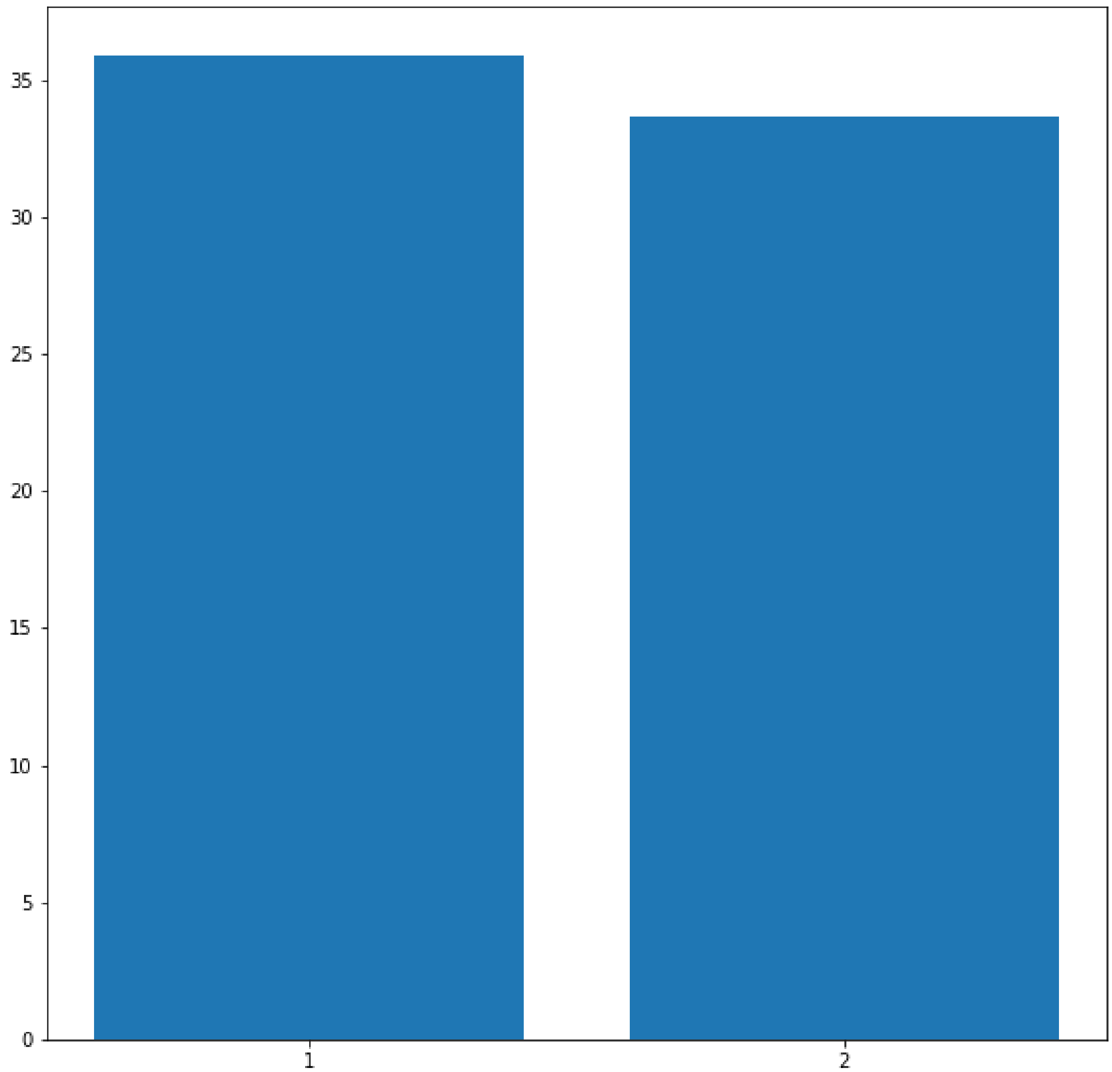
[148]:
```
Y = df.iloc[:, -1]
Y
```

[148]:
```
0      39.0
1      81.0
2       6.0
3      77.0
4      40.0
       …
195    79.0
196    28.0
```

```
197    74.0
198    18.0
199    83.0
Name: Spending Score (1-100), Length: 200, dtype: float64
```

[149]:
```
pca = PCA(2)
data = pca.fit_transform(x)
```

[150]:
```
plt.figure(figsize=(10,10))
var = np.round(pca.explained_variance_ratio_*100, decimals = 1)
lbls = [str(x) for x in range(1,len(var)+1)]
plt.bar(x=range(1,len(var)+1), height = var, tick_label = lbls)
plt.show()
```



```
[158]: X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.
        ↪20,random_state=42)
```

Build the Model

```
[159]: #Importing KMeans from sklearn
       from sklearn.cluster import KMeans
       import matplotlib.pyplot as plt
```

```
[160]: wcss=[]
       for i in range(1,11):
           km=KMeans(n_clusters=i)
           km.fit(X)
           wcss.append(km.inertia_)
```

```
[ ]: #The elbow curve
     plt.figure(figsize=(12,6))
     plt.plot(range(1,11),wcss)
     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()
```

```
[154]: #Taking 5 clusters
       km1=KMeans(n_clusters=4)
       #Fitting the input data
       km1.fit(X_train)
       #predicting the labels of the input data
       y=km1.predict(X_test)
       #adding the labels to a column named label
       df["label"] = y
       #The new dataframe with the clustering done
       df.head()
```

[154]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	label
0	1	19.0	15.0	39.0	2
1	1	21.0	15.0	81.0	2
2	0	20.0	16.0	6.0	2
3	0	23.0	16.0	77.0	2
4	0	31.0	17.0	40.0	2

```
[155]: #Scatterplot of the clusters
       plt.figure(figsize=(10,6))
       sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score␣
        ↪(1-100)',hue="label",
```

```
palette=['green','orange','brown', 'blue'], legend='full',data␣
↪= df  ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```