CUSTOMER SEGMENTATION ANALYSIS
Assignment -4

Assignment Date	26 October 2022
Team ID	PNT2022TMID27812
Project Name	Smart Lender-Application Credibility Prediction for loan Approval
Student Name	S.G.Mydhrayan
Student Roll Number	311519104036
Maximum Marks	2 Marks

Question-1. Download dataset

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

Question-2. Load the dataset

Solution:

```
df = pd.read_csv('Mall_Customers.csv')
df = df.drop(columns=["CustomerID"])
df.head()
```

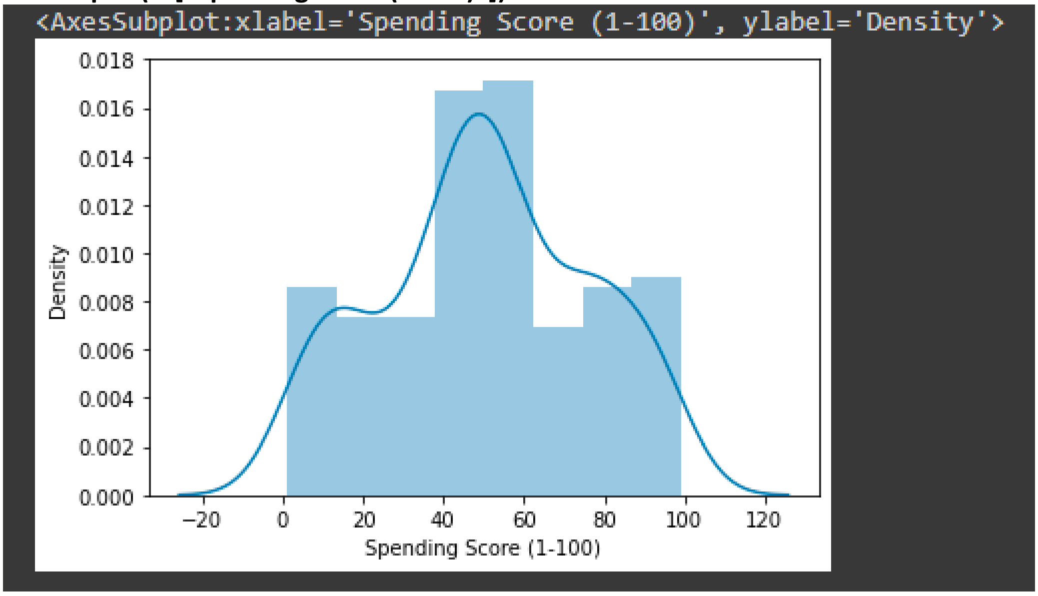
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

Question-3. Perform Below Visualizations.

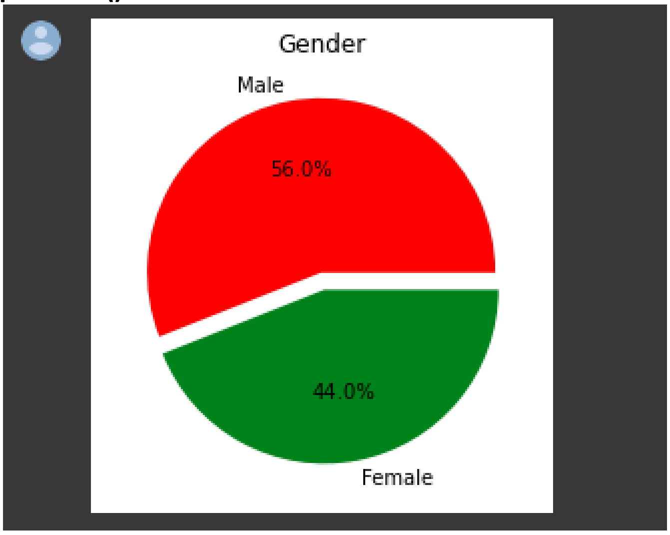
3.1 Univariate Analysis

Solution:

sns.distplot(df["Spending Score (1-100)"])



plt.pie(df.Gender.value_counts(),[0.05,0.05],colors=['red','green'],labels=['Male','Female'],autopct ="%1.1f%%")
plt.title('Gender')
plt.show()

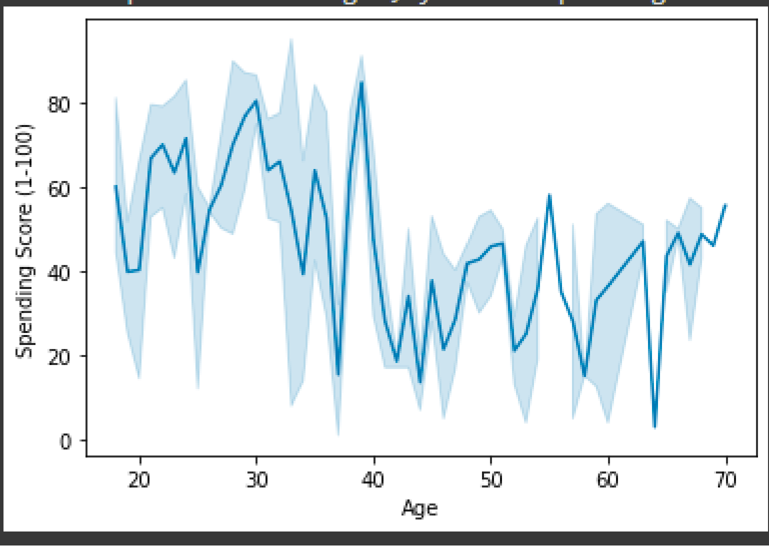


3.2 Bivariate Analysis

Solution:

sns.lineplot(df['Age'],df["Spending Score (1-100)"])

```
<AxesSubplot:xlabel='Age', ylabel='Spending Score (1-100)'>
```

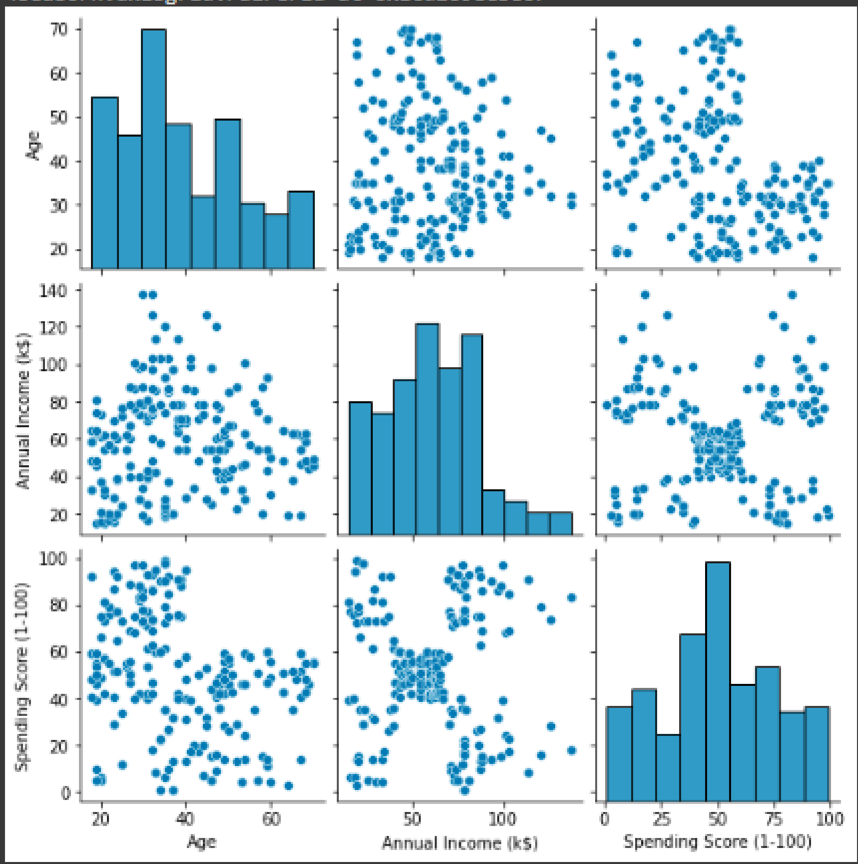


3.3 Multivariate Analysis

Solution:

sns.pairplot(df)

```
&lt;seaborn.axisgrid.PairGrid at 0x1cd269eebe0&gt;
```



df.corr()

	Age	Annual Income (k$)	Spending Score (1-100)
Age	1.000000	-0.012398	-0.327227
Annual Income (k$)	-0.012398	1.000000	0.009903
Spending Score (1-100)	-0.327227	0.009903	1.000000

sns.heatmap(df.corr(),annot=True)

plt.show()

Question-4. Perform descriptive statistics on the dataset.

Solution:

df.describe()

	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

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Gender                  200 non-null    object
 1   Age                     200 non-null    int64 
 2   Annual Income (k$)      200 non-null    int64 
 3   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 6.4+ KB
```

Question-5. Check for Missing values and deal with them.

Solution:

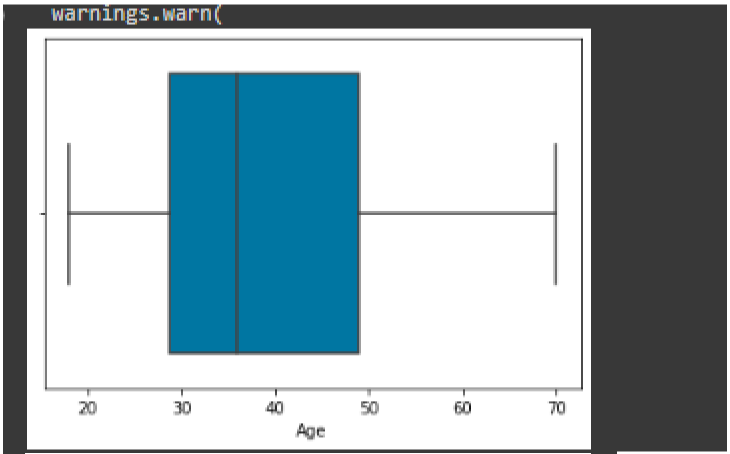
df.isnull().sum()

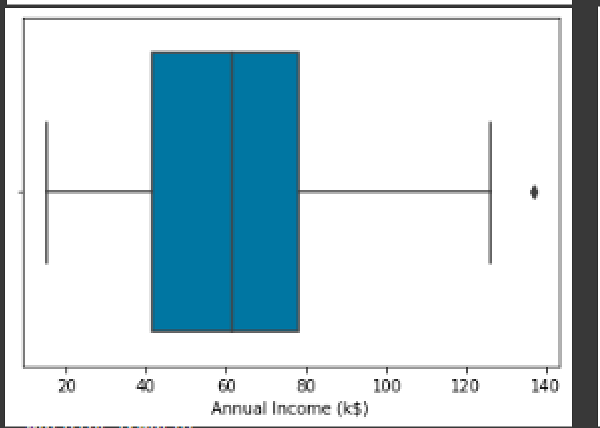
```
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

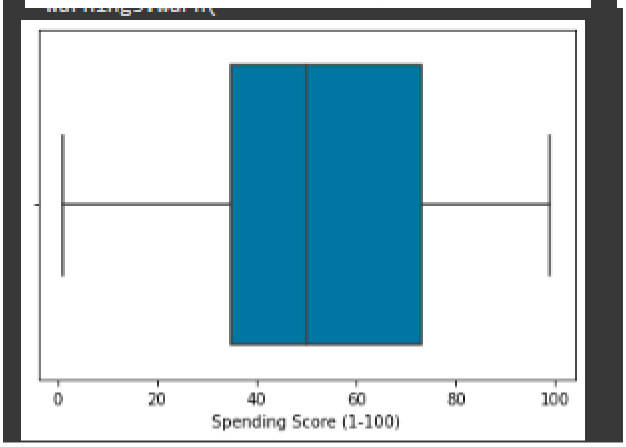
Question-6. Find the outliers and replace the outliers

Solution:

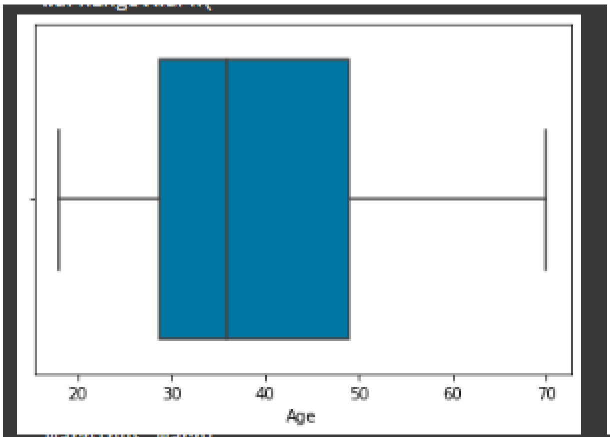
```
for i in df.columns.drop("Gender"):
  sns.boxplot(df[i])
  plt.show()
```

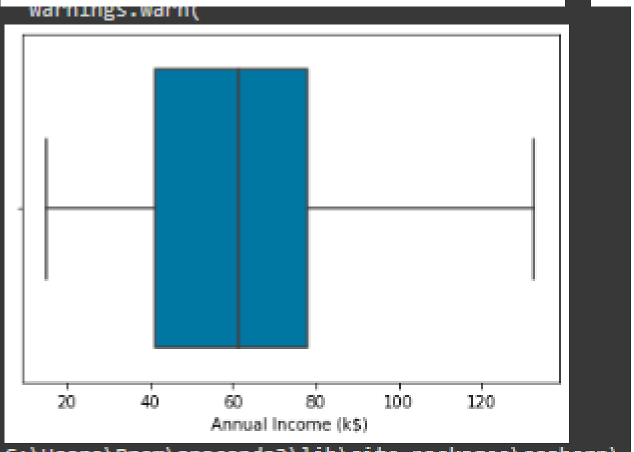


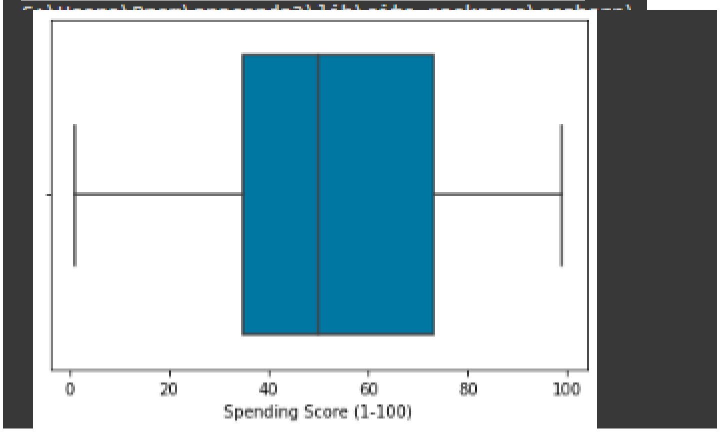





```
for i in df.columns.drop('Gender'):
    Q1 = df[i].quantile(0.25)
    Q3 = df[i].quantile(0.75)
    IQR = Q3-Q1
    upper_limit = Q3 + (1.5*IQR)
    lower_limit = Q1 - (1.5*IQR)
    df[i] = np.where(df[i]>=upper_limit,Q3 + (1.5*IQR),df[i])
    df[i] = np.where(df[i]<=lower_limit,Q1 - (1.5*IQR),df[i])
for i in  df.columns.drop('Gender'):
    sns.boxplot(df[i])
    plt.show()
```







Question-7. Check for Categorical columns and perform encoding

Solution:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df.Gender = le.fit_transform(df.Gender)
df.head()
```

	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

Question-8. Scaling the

Solution:

```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
df = pd.DataFrame(scale.fit_transform(df),columns=df.columns)
df.head()
```

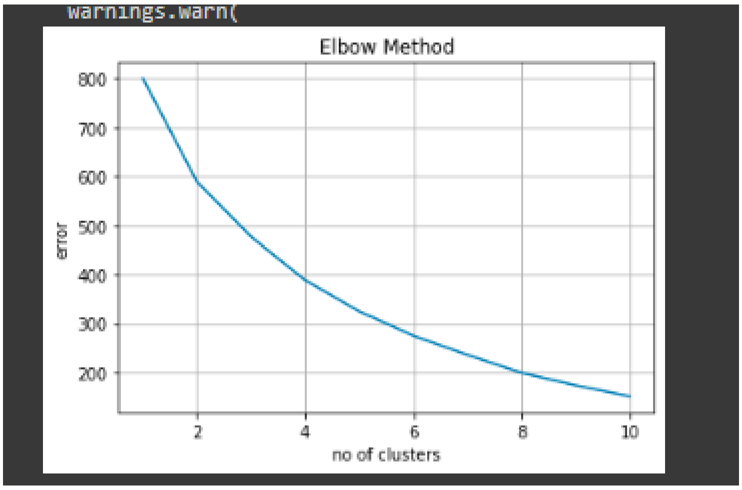
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.128152	-1.424569	-1.745429	-0.434801
1	1.128152	-1.281035	-1.745429	1.195704
2	-0.886405	-1.352802	-1.707083	-1.715913
3	-0.886405	-1.137502	-1.707083	1.040418
4	-0.886405	-0.563369	-1.668737	-0.395980

Question-9. Perform any of the clustering algorithms

Solution:

```
from sklearn.cluster import KMeans
error = []
for k in range(1,11):
  kmeans = KMeans(n_clusters=k,init='k-means++')
  kmeans.fit(df)
  error.append(kmeans.inertia_)
```

plt.plot(range(1,11),error)
plt.title('Elbow Method')
plt.xlabel('no of clusters')
plt.ylabel('error')
plt.grid()
plt.show()



km = KMeans(n_clusters=8)
Category = km.fit_predict(df)
Category

```
array([5, 5, 7, 7, 7, 7, 0, 7, 4, 7, 4, 7, 0, 7, 5, 5, 7, 5, 4, 7, 5, 5,
       0, 5, 0, 5, 0, 5, 0, 7, 4, 7, 4, 5, 0, 7, 0, 7, 0, 7, 0, 5, 4, 7,
       0, 7, 0, 7, 7, 7, 0, 5, 7, 4, 0, 4, 0, 4, 7, 4, 4, 5, 0, 0, 4, 5,
       0, 0, 5, 7, 4, 0, 0, 0, 4, 5, 0, 5, 7, 0, 4, 5, 4, 0, 7, 4, 0, 7,
       7, 0, 0, 5, 4, 0, 7, 5, 0, 7, 4, 5, 7, 0, 4, 5, 4, 7, 0, 4, 4, 4,
       4, 7, 6, 5, 7, 7, 0, 0, 0, 0, 5, 6, 1, 2, 6, 1, 3, 2, 4, 2, 3, 2,
       6, 1, 3, 1, 6, 2, 3, 1, 6, 2, 6, 1, 3, 2, 3, 1, 6, 2, 3, 2, 6, 1,
       6, 1, 3, 1, 3, 1, 6, 1, 3, 1, 3, 1, 3, 1, 6, 2, 3, 2, 3, 2, 6, 1,
       3, 2, 3, 2, 6, 1, 3, 1, 6, 2, 6, 2, 6, 1, 6, 1, 3, 1, 6, 1, 6, 2,
       3, 2])
```

Question-10. Add the cluster data with the primary dataset

Solution:

```
df["Category"] = pd.Series(Category)
df.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Category
0	1.128152	-1.424569	-1.745429	-0.434801	5
1	1.128152	-1.281035	-1.745429	1.195704	5
2	-0.886405	-1.352802	-1.707083	-1.715913	7
3	-0.886405	-1.137502	-1.707083	1.040418	7
4	-0.886405	-0.563369	-1.668737	-0.395980	7

Question-11. Add the cluster data with the primary dataset

Solution:

```
X = df.drop(columns=["Category"])
Y = df.Category
```

Question-12. Add the cluster data with the primary dataset

Solution:

```
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(X,Y,test_size=0.
2,random_state=0)
```

Question-13. Add the cluster data with the primary dataset

Solution:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
```

Question-14. Add the cluster data with the primary dataset

Solution:

model.fit(x_train,y_train)

```
RandomForestClassifier()
```

Question-15. Add the cluster data with the primary dataset

Solution:

y_predict = model.predict(x_test)
pd.DataFrame({"Actual":y_test,"Predicted":y_predict.round(0)})

	Actual	Predicted
18	4	4
170	3	3
107	4	4
98	4	4
177	2	2
182	3	3
5	7	7
146	3	3
12	0	0
152	6	6
61	5	5
125	1	1
180	6	6
154	6	6

80	4	4
7	7	7
33	5	5
130	3	3
37	7	7
74	4	4
183	1	1
145	2	2
45	7	7
159	1	1
60	4	4
123	2	2
179	2	2
185	2	2
122	1	0
44	0	0
16	7	0
55	4	4
150	3	3
111	7	7
22	0	0
189	1	1
129	2	2
4	7	7
83	0	0
106	0	0

Question-16. Add the cluster data with the primary dataset

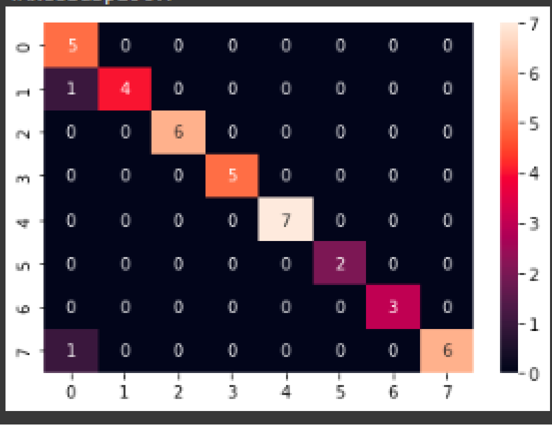
Solution:

from sklearn import metrics
metrics.accuracy_score(y_test,y_predict)

```
0.95
```

sns.heatmap(metrics.confusion_matrix(y_test,y_predict),annot=True)

```
<AxesSubplot:>
```



print(metrics.classification_report(y_test,y_predict))

```
              precision    recall  f1-score   support

           0       0.71      1.00      0.83         5
           1       1.00      0.80      0.89         5
           2       1.00      1.00      1.00         6
           3       1.00      1.00      1.00         5
           4       1.00      1.00      1.00         7
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         3
           7       1.00      0.86      0.92         7

    accuracy                           0.95        40
   macro avg       0.96      0.96      0.96        40
weighted avg       0.96      0.95      0.95        40
```