

# UNIVERSITY ADMIT ELIGIBILITY PREDICTOR

## ASSIGNMENT - 4

Date	27th October 2022
Team ID	PNT2022TMID54388
Student Name	MOSIKARAN M (310619106083)
Domain Name	Education
Project Name	University Admit Eligibility Predictor
Maximum Marks	2 Marks

### 1.)IMPORT THE REQUIRED LIBRARIES

#### 1.)IMPORT THE REQUIRED LIBRARIES

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

### 2.)DOWNLOAD AND UPLOAD THE DATASET

#### 2.)DOWNLOAD AND UPLOAD THE DATASET INTO THE TOOL

```
In [2]: df = pd.read_csv('Mall_Customers.csv')
df = df.drop(columns=["CustomerID"])
df.head()
```

```
Out[2]:
```

	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

### 3.)CHECK FOR MISSING VALUES AND DEAL WITH THEM

#### 3.)CHECK FOR MISSING VALUES AND DEAL WITH THEM

```
In [3]: df.isnull().sum()
```

```
Out[3]: Gender          0  
Age          0  
Annual Income (k$)      0  
Spending Score (1-100)  0  
dtype: int64
```

### 4.) PERFORM THE DESCRIPTIVE STATISTICS ON THE DATASET

#### 4.)PERFORM DESCRIPTIVE STATISTICS ON THE DATASET

```
In [4]: df.describe()
```

```
Out[4]:
```

	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

```
In [5]: 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
```

## 5.) PERFORM VARIOUS VISUALISATIONS

### a.) UNIVARIANTE ANALYSIS

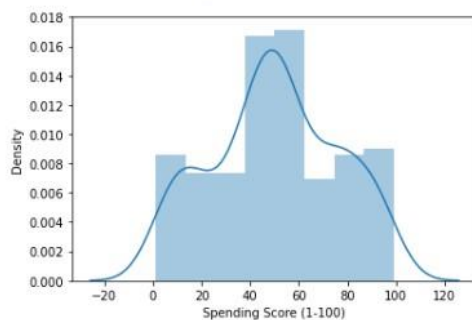
#### 5.)PERFORM VISUALIZATIONS

#### a.)UNIVARIANTE ANALYSIS

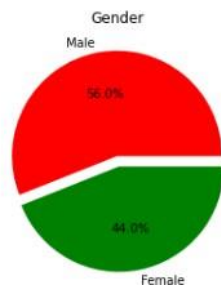
```
In [6]: sns.distplot(df["Spending Score (1-100)"])
```

C:\Users\Prem\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
warnings.warn(msg, FutureWarning)

```
Out[6]: <AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Density'>
```



```
In [7]: plt.pie(df.Gender.value_counts(),[0.05,0.05],colors=['red','green'],labels=['Male','Female'],autopct="%1.1f%%")  
plt.title('Gender')  
plt.show()
```



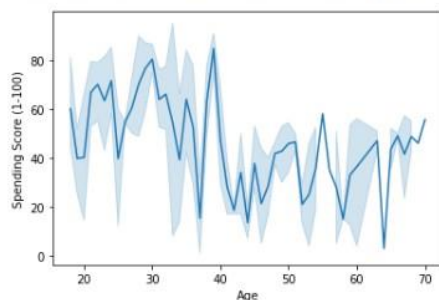
### b.) BI - VARIANTE ANALYSIS

#### b.)BI - VARIANTE ANALYSIS

```
In [8]: sns.lineplot(df['Age'],df["Spending Score (1-100)"])
```

C:\Users\Prem\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
warnings.warn()

```
Out[8]: <AxesSubplot:xlabel='Age', ylabel='Spending Score (1-100)'>
```

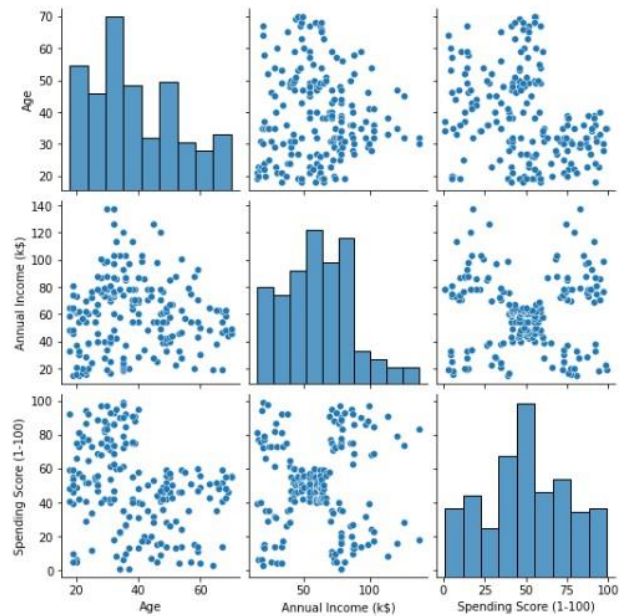


## c.) MULTI - VARIANTE ANALYSIS

### c.)MULTIVARIANTE ANALYSIS

```
In [9]: sns.pairplot(df)
```

```
Out[9]: <seaborn.axisgrid.PairGrid at 0x1cd269eebe0>
```



```
In [10]: df.corr()
```

```
Out[10]:
```

	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

```
In [11]: sns.heatmap(df.corr(),annot=True)  
plt.show()
```

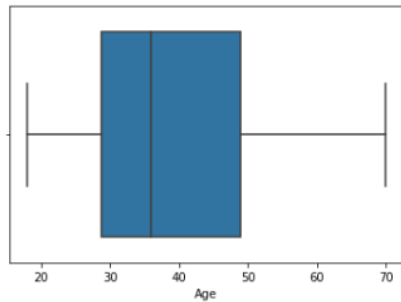


## 6.) FIND AND REPLACE THE OUTLIERS

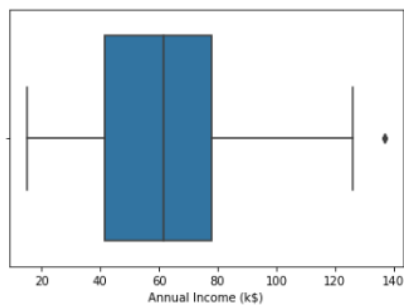
### 6.)FIND THE OUTLIERS AND REPLACE THE OUTLIERS

```
In [12]: for i in df.columns.drop("Gender"):  
sns.boxplot(df[i])  
plt.show()
```

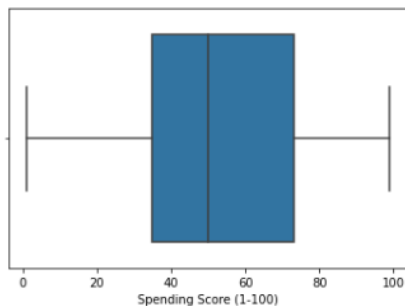
C:\Users\Prem\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword argument: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key word will result in an error or misinterpretation.  
warnings.warn()



C:\Users\Prem\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword argument: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key word will result in an error or misinterpretation.  
warnings.warn()



C:\Users\Prem\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword argument: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key word will result in an error or misinterpretation.  
warnings.warn()

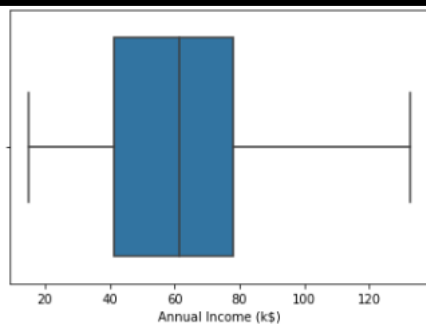
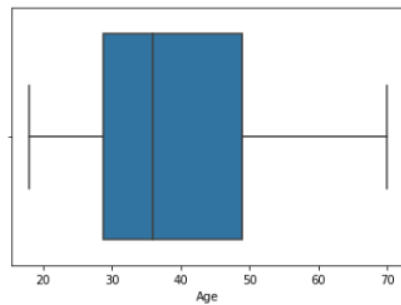


```
In [13]: for i in df.columns.drop('Gender'):
        Q1 = df[i].quantile(0.25)
        Q3 = df[i].quantile(0.75)
        IQR = Q3 - Q1
        upper_limit = Q3 + (1.5*IQR)
        lower_limit = Q1 - (1.5*IQR)
        df[i] = np.where(df[i]>upper_limit,Q3 + (1.5*IQR),df[i])
        df[i] = np.where(df[i]<lower_limit,Q1 - (1.5*IQR),df[i])
```

```
In [14]: for i in df.columns.drop('Gender'):
        sns.boxplot(df[i])
        plt.show()
```

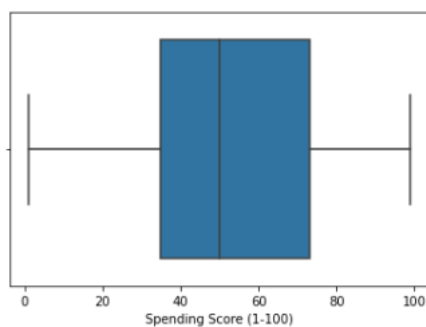
C:\Users\Prem\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword argument: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



C:\Users\Prem\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variable as a keyword argument: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(



## 7.)CHECKFORCATEGORICALCOLUMNSANDENCODETHEM

### 7.)CHECK FOR CATEGORICAL COLUMNS AND PERFORM ENCODING

```
In [15]: from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
df.Gender = le.fit_transform(df.Gender)
```

```
In [16]: df.head()
```

```
Out[16]:
```

	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

## 8.) SCALE THE DATA

### 8.)SCALING THE DATA

```
In [17]: from sklearn.preprocessing import StandardScaler  
scale = StandardScaler()  
df = pd.DataFrame(scale.fit_transform(df), columns=df.columns)  
df.head()
```

```
Out[17]:
```

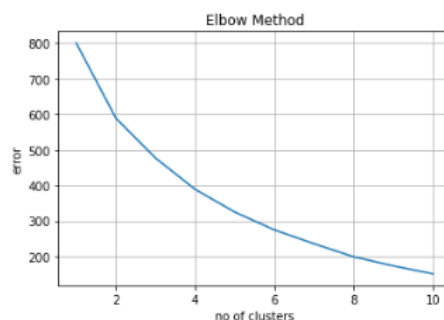
	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1.128152	-1.424569	-1.745429	-0.434801
1	1.128152	-1.281035	-1.745429	1.195704
2	-0.886405	-1.352802	-1.707083	-1.715913
3	-0.886405	-1.137502	-1.707083	1.040418
4	-0.886405	-0.563369	-1.668737	-0.395980

## 9.)PERFORMANYOFTHECLUSTERINGALGORITHMS

### 9.)PERFORM ANY OF THE CLUSTERING ALGORITHMS

```
In [18]: 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()
```

C:\Users\Prem\anaconda3\lib\site-packages\sklearn\cluster\\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP\_NUM\_THREADS=1.  
warnings.warn(



```
In [19]: km = KMeans(n_clusters=8)
Category = km.fit_predict(df)
Category
```

```
Out[19]: 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, 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, 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])
```

## 10.) ADDING THE CLUSTER WITH THE PRIMARY DATASET

### 10.)ADD THE CLUSTER DATA WITH THE PRIMARY DATASET

```
In [20]: df["Category"] = pd.Series(Category)
df.head()
```

```
Out[20]:
```

	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	1.196704	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



## 11.) SPLITTING THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

### 11.)SPLITTING THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

```
In [21]: X = df.drop(columns=["Category"])
        Y = df.Category
```

## 12.) SPLIT THE DATA INTO TRAINING AND TESTING DATA

### 12.)SPLIT THE DATA INTO TRAINING AND TESTING DATA

```
In [22]: 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)
```

## 13.) BUILD THE MODEL

### 13.)BUILD THE MODEL

```
In [23]: from sklearn.ensemble import RandomForestClassifier
        model = RandomForestClassifier()
```

## 14.) TRAIN THE MODEL

### 14.)TRAIN THE MODEL

```
In [24]: model.fit(x_train,y_train)

Out[24]: RandomForestClassifier()
```

## 15.) TEST THE MODEL

### 15.)TEST THE MODEL

```
In [25]: y_predict = model.predict(x_test)
```

```
In [26]: pd.DataFrame({"Actual":y_test,"Predicted":y_predict.round(0)})
```

Out[26]:

	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

## 16.) MEASURE THE PERFORMANCE USING METRICS

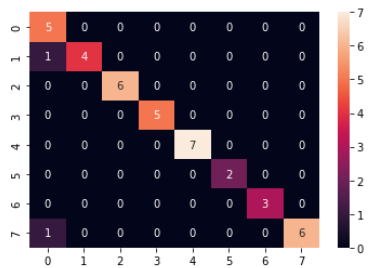
### 16.)MEASURE THE PERFORMANCE USING METRICS

```
In [27]: from sklearn import metrics  
metrics.accuracy_score(y_test,y_predict)
```

```
Out[27]: 0.95
```

```
In [28]: sns.heatmap(metrics.confusion_matrix(y_test,y_predict),annot=True)
```

```
Out[28]: <AxesSubplot:>
```



```
In [29]: 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