Assignment -5

Python Programming

Assignment Date	09 November 2022
Student Name	S.Yuvaraj Sai
Student Roll Number	210519205060
Maximum Marks	2 Marks

Assignment-4

Customer Segmentation Analysis

Double-click (or enter) to edit

Problem Statement:-

You own the mall and want to understand the customers who can quickly converge [Target Customers] so that the insight can be given to the marketing team and plan the strategy accordingly.

Clustering the data and performing classification algorithms

1. Perform Below Visualizations.

Import Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
```

Import Dataset

```
data = pd.read_csv('Mall_Customers.csv')
data
```

	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
...	...	...	...	...	...
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

200 rows × 5 columns

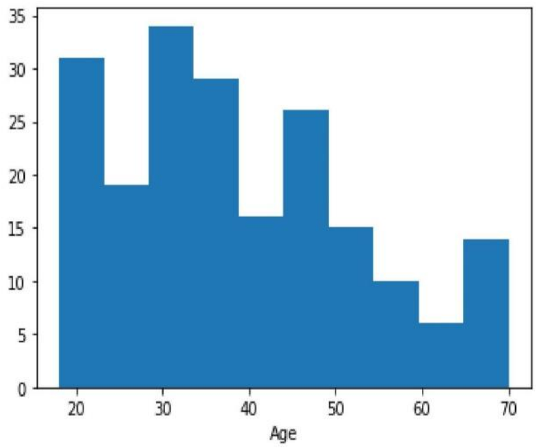
```
data.info()
```

```
<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
```

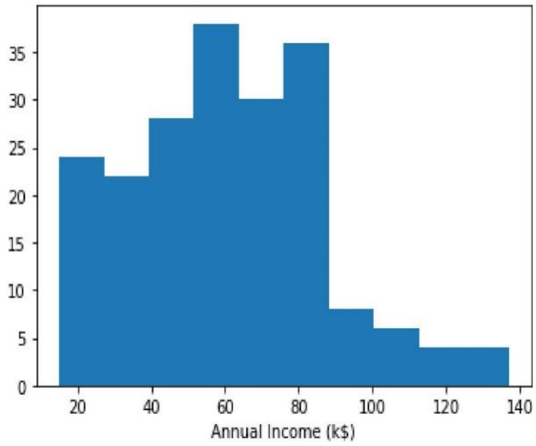
Univariate Analysis

```
plt.hist(data['Gender']);
plt.xlabel('Gender');
```

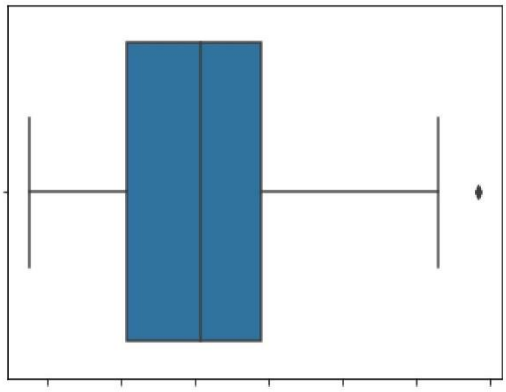

```
plt.hist(data['Age']);
plt.xlabel('Age');
```



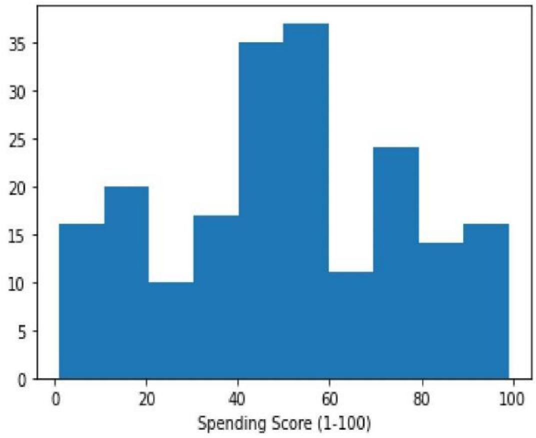

```
plt.hist(data['Annual Income (k$)']);
plt.xlabel('Annual Income (k$)');
```




```
sns.boxplot(x=data['Annual Income (k$)'])
plt.xlabel('Annual Income (k$)');
```

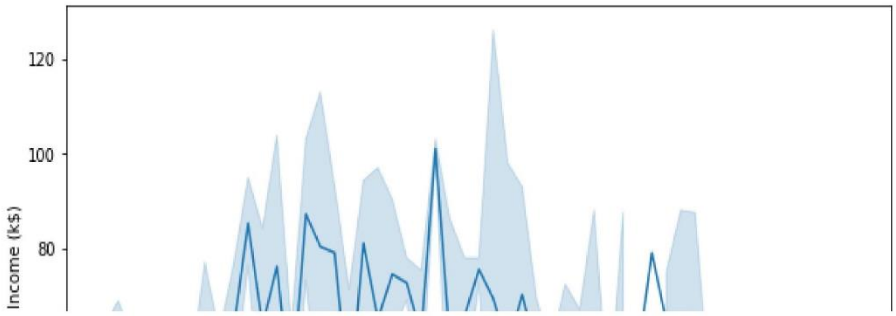


```
plt.hist(data['Spending Score (1-100)']);
plt.xlabel('Spending Score (1-100)');
```

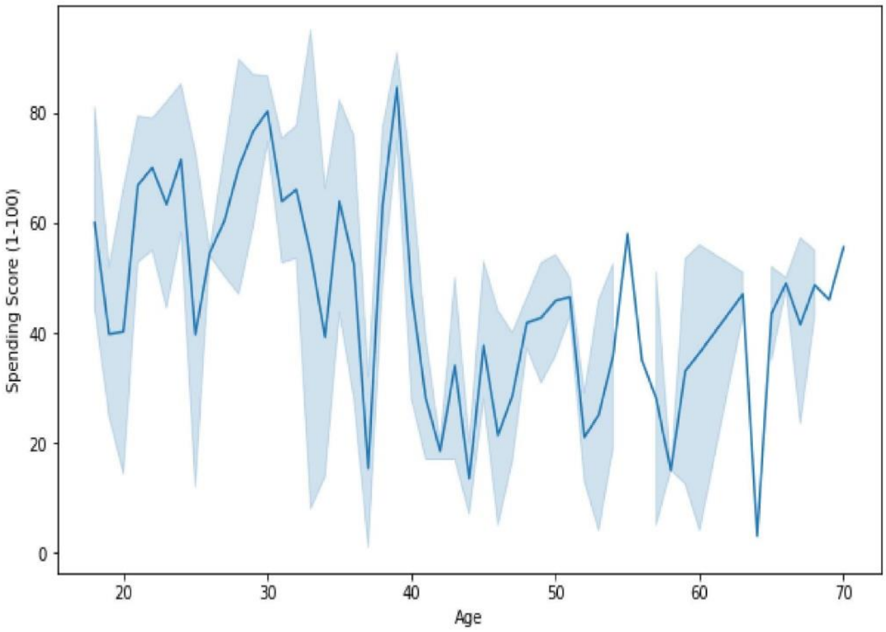


Bivariate Analysis

```
plt.figure(figsize=(10, 6))
sns.lineplot(x=data["Age"], y=data["Annual Income (k$)"]);
plt.xlabel('Age');
plt.ylabel('Annual Income (k$)');
```

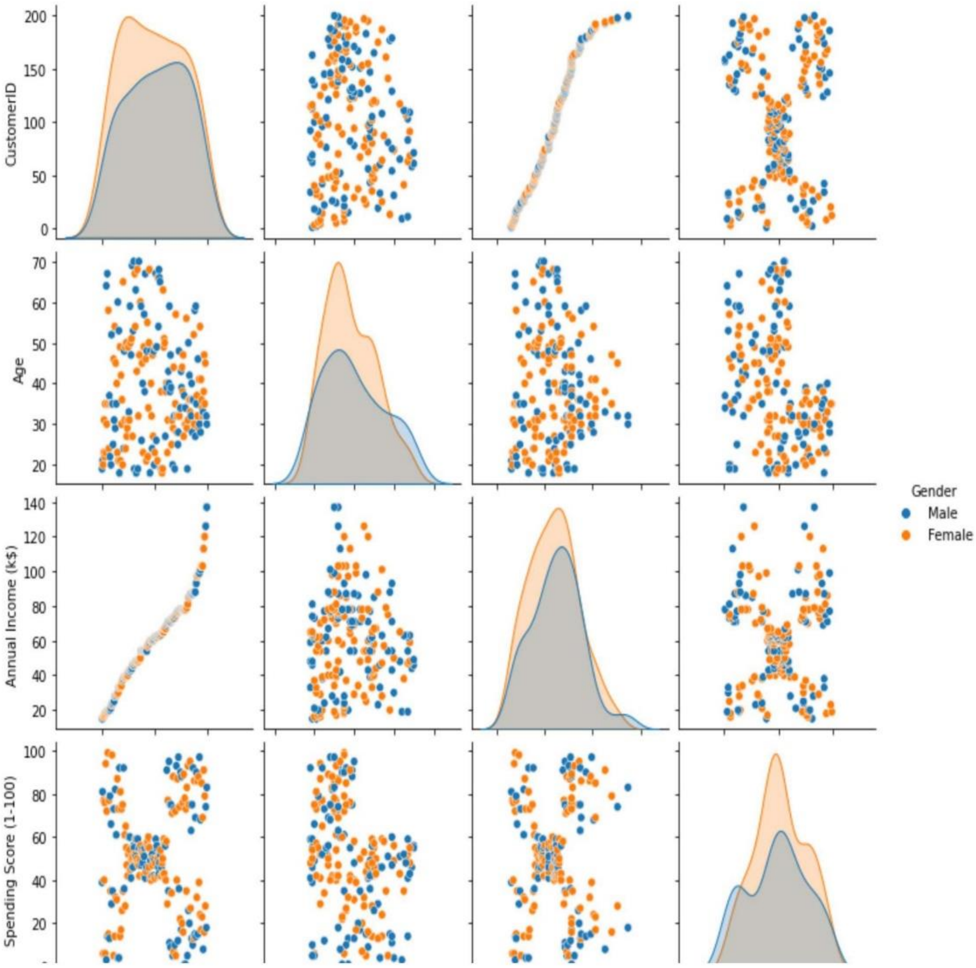



```
plt.figure(figsize=(10, 6))
sns.lineplot(x=data["Age"], y=data["Spending Score (1-100)"]);
plt.xlabel('Age');
plt.ylabel('Spending Score (1-100)');
```



Multi-variate Analysis

```
sns.pairplot(data, hue='Gender');
```




```
plt.figure(figsize=(10, 6));
sns.heatmap(data.corr(), annot=True);
```

Descriptive Statistics


```
data.describe()
```

	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

```
data.skew()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
CustomerID                0.000000
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

```
data.kurt()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
CustomerID               -1.200000
Age                      -0.671573
Annual Income (k$)       -0.098487
Spending Score (1-100)   -0.826629
dtype: float64
```

```
data.var()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
```

```
CustomerID                3350.000000
Age                        195.133166
Annual Income (k$)         689.835578
Spending Score (1-100)     666.854271
dtype: float64
```

Handling Missing Values

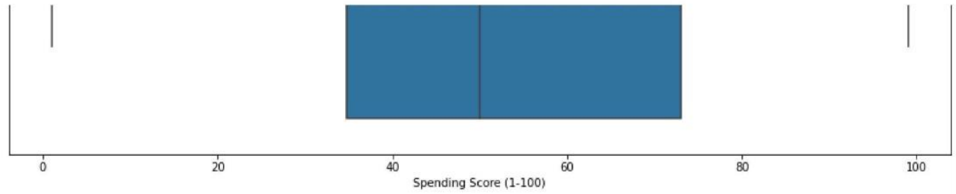
```
data.isna().sum()
```

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

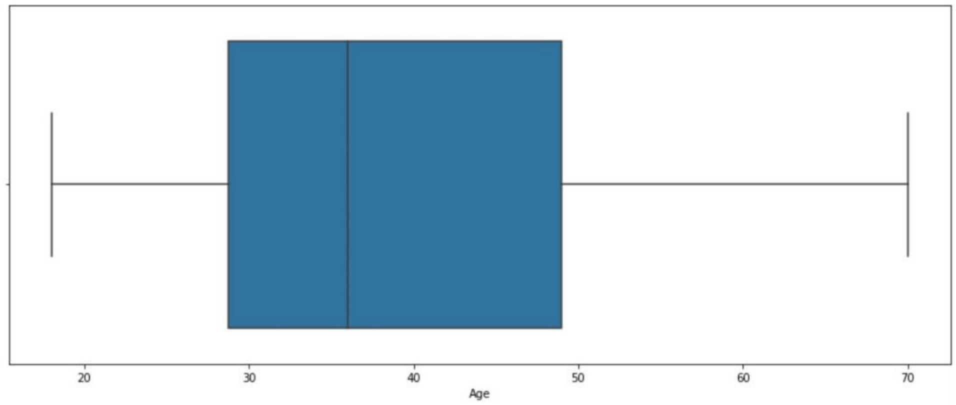
Outlier Handling

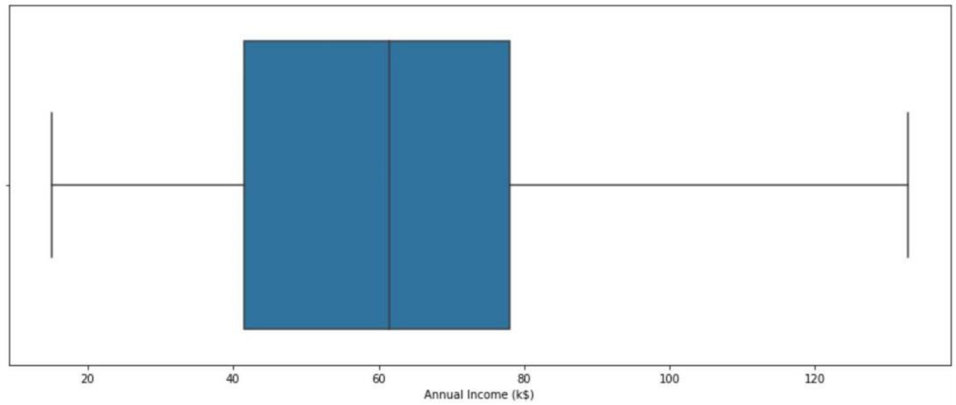
```
numeric_cols = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']

def boxplots(cols):
    fig, axes = plt.subplots(3, 1, figsize=(15, 20))

    t=0
    for i in range(3):
        sns.boxplot(ax=axes[i], data=data, x=cols[t])
        t+=1

    plt.show()


def Flooring_outlier(col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    whisker_width = 1.5
    lower_whisker = Q1 -(whisker_width*IQR)
    upper_whisker = Q3 + (whisker_width*IQR)
    data[col]=np.where(data[col]>upper_whisker,upper_whisker,np.where(data[col]<lower_whisker


print('Before Outliers Handling')
print('='*100)
boxplots(numeric_cols)
for col in numeric_cols:
```

```
        Flooring_outlier(col)
    print('\n\n\nAfter Outliers Handling')
    print('='*100)
    boxplots(numeric_cols)
```




```
After Outliers Handling
==========================================================================================
```





Encode Categorical Columns

```
data = pd.get_dummies(data, columns = ['Gender'])
data
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	1	19.0	15.00	39.0	0	1
1	2	21.0	15.00	81.0	0	1
2	3	20.0	16.00	6.0	1	0

Standard Scaling

```
data = data.drop(['CustomerID'], axis=1)
data
```

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	19.0	15.00	39.0	0	1
1	21.0	15.00	81.0	0	1
2	20.0	16.00	6.0	1	0
3	23.0	16.00	77.0	1	0
4	31.0	17.00	40.0	1	0
...	...	...	...	...	...
195	35.0	120.00	79.0	1	0
196	45.0	126.00	28.0	1	0
197	32.0	126.00	74.0	0	1
198	32.0	132.75	18.0	0	1
199	30.0	132.75	83.0	0	1

200 rows × 5 columns

```
cols = data.columns
cols
```

```
Index(['Age', 'Annual Income (k$)', 'Spending Score (1-100)', 'Gender_Female',
       'Gender_Male'],
      dtype='object')
```

```
scaler = StandardScaler()
sc = scaler.fit_transform(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']])
sc
```

```
       [-0.49160182,  0.55535083,  1.6615628 ],
       [-0.77866858,  0.59369717, -0.39597992],
       [-0.49160182,  0.59369717,  1.42863343]
```

```
[-0.99396865,  0.6320435 , -1.48298362],
[-0.77866858,  0.6320435 ,  1.81684904],
[ 0.65666521,  0.6320435 , -0.55126616],
[-0.49160182,  0.6320435 ,  0.92395314],
[-0.34806844,  0.67038984, -1.09476801],
[-0.34806844,  0.67038984,  1.54509812],
[ 0.29783176,  0.67038984, -1.28887582],
[ 0.010765  ,  0.67038984,  1.46745499],
[ 0.36959845,  0.67038984, -1.17241113],
[-0.06100169,  0.67038984,  1.00159627],
[ 0.58489852,  0.67038984, -1.32769738],
[-0.85043527,  0.67038984,  1.50627656],
[-0.13276838,  0.67038984, -1.91002079],
[-0.6351352 ,  0.67038984,  1.07923939],
[-0.34806844,  0.67038984, -1.91002079],
[-0.6351352 ,  0.67038984,  0.88513158],
[ 1.23079873,  0.70873618, -0.59008772],
[-0.70690189,  0.70873618,  1.27334719],
[-1.42456879,  0.78542885, -1.75473454],
[-0.56336851,  0.78542885,  1.6615628 ],
[ 0.80019859,  0.9388142 , -0.93948177],
[-0.20453507,  0.9388142 ,  0.96277471],
[ 0.22606507,  0.97716054, -1.17241113],
[-0.41983513,  0.97716054,  1.73920592],
[-0.20453507,  1.01550688, -0.90066021],
[-0.49160182,  1.01550688,  0.49691598],
[ 0.08253169,  1.01550688, -1.44416206],
[-0.77866858,  1.01550688,  0.96277471],
[-0.20453507,  1.01550688, -1.56062674],
[-0.20453507,  1.01550688,  1.62274124],
[ 0.94373197,  1.05385321, -1.44416206],
[-0.6351352 ,  1.05385321,  1.38981187],
[ 1.37433211,  1.05385321, -1.36651894],
[-0.85043527,  1.05385321,  0.72984534],
[ 1.4460988 ,  1.2455849 , -1.4053405 ],
[-0.27630176,  1.2455849 ,  1.54509812],
[-0.13276838,  1.39897025, -0.7065524 ],
[-0.49160182,  1.39897025,  1.38981187],
[ 0.51313183,  1.43731659, -1.36651894],
[-0.70690189,  1.43731659,  1.46745499],
[ 0.15429838,  1.47566292, -0.43480148],
[-0.6351352 ,  1.47566292,  1.81684904],
[ 1.08726535,  1.5523556 , -1.01712489],
[-0.77866858,  1.5523556 ,  0.69102378],
[ 0.15429838,  1.62904827, -1.28887582],
[-0.20453507,  1.62904827,  1.35099031],
[-0.34806844,  1.62904827, -1.05594645],
[-0.49160182,  1.62904827,  0.72984534],
[-0.41983513,  2.01251165, -1.63826986],
[-0.06100169,  2.01251165,  1.58391968],
[ 0.58489852,  2.28093601, -1.32769738],
[-0.27630176,  2.28093601,  1.11806095],
[ 0.44136514,  2.51101403, -0.86183865],
[-0.49160182,  2.51101403,  0.92395314],
[-0.49160182,  2.76985181, -1.25005425],
```

```
data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = sc
data
```

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	-1.424569	-1.745429	-0.434801	0	1
1	-1.281035	-1.745429	1.195704	0	1
2	-1.352802	-1.707083	-1.715913	1	0
3	-1.137502	-1.707083	1.040418	1	0
4	-0.563369	-1.668737	-0.395980	1	0
...	...	...	...	...	...
195	-0.276302	2.280936	1.118061	1	0
196	0.441365	2.511014	-0.861839	1	0
197	-0.491602	2.511014	0.923953	0	1
198	-0.491602	2.769852	-1.250054	0	1
199	-0.635135	2.769852	1.273347	0	1

200 rows × 5 columns

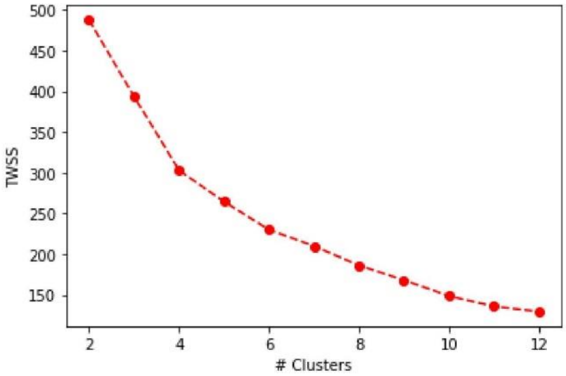
Clustering

```
TWSS = []
k = list(range(2,13))

for i in k:
    kmeans = KMeans(n_clusters = i , init = 'k-means++')
    kmeans.fit(data)
    TWSS.append(kmeans.inertia_)

TWSS
```

```
[487.65867172744953,
 393.6497829986831,
 302.7542334541679,
 264.3903755157514,
 229.9106024423537,
 209.77702386241236,
 186.0163645303913,
 167.9777361984937,
 148.39231267577674,
 135.69562671961853,
 129.30659450120024]
```

```
plt.plot(k, TWSS, 'ro--')
plt.xlabel('# Clusters')
plt.ylabel('TWSS')
```

```
Text(0, 0.5, 'TWSS')
```




```
model = KMeans(n_clusters = 5)
model.fit(data)
```

```
KMeans(n_clusters=5)
```

Add the Cluster data with Primary dataset

```
mb = pd.Series(model.labels_)
data['Cluster'] = mb
data
```

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male	Cluster
0	-1.424569	-1.745429	-0.434801	0	1	0
1	-1.281035	-1.745429	1.195704	0	1	0

```
data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = scaler.inverse_transform(data
data
```

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male	Cluster
0	19.0	15.00	39.0	0	1	0
1	21.0	15.00	81.0	0	1	0
2	20.0	16.00	6.0	1	0	3
3	23.0	16.00	77.0	1	0	0
4	31.0	17.00	40.0	1	0	3
...	...	...	...	...	...	...
195	35.0	120.00	79.0	1	0	4
196	45.0	126.00	28.0	1	0	1
197	32.0	126.00	74.0	0	1	4
198	32.0	132.75	18.0	0	1	1
199	30.0	132.75	83.0	0	1	4

200 rows × 6 columns

```
mb=pd.Series(model.labels_)
data
```

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male	Cluster
0	19.0	15.00	39.0	0	1	0
1	21.0	15.00	81.0	0	1	0

Split Data Into Dependent & Independent Features

```
X=data.drop('Cluster',axis=1)
Y=data['Cluster']
X, Y
```

```
(      Age  Annual Income (k$)  Spending Score (1-100)  Gender_Female  \
 0    19.0               15.00                    39.0              0
 1    21.0               15.00                    81.0              0
 2    20.0               16.00                     6.0              1
 3    23.0               16.00                    77.0              1
 4    31.0               17.00                    40.0              1
 ..    ...                 ...                     ...            ...
 195  35.0              120.00                    79.0              1
 196  45.0              126.00                    28.0              1
 197  32.0              126.00                    74.0              0
 198  32.0              132.75                    18.0              0
 199  30.0              132.75                    83.0              0

      Gender_Male
 0              1
 1              1
 2              0
 3              0
 4              0
 ..           ...
 195            0
 196            0
 197            1
 198            1
 199            1

 [200 rows x 5 columns], 0      0
 1      0
 2      3
 3      0
 4      3
       ..
 195    4
 196    1
 197    4
 198    1
 199    4
 Name: Cluster, Length: 200, dtype: int32)
```

Split the data into Training And Testing Data

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

```
((160, 5), (40, 5), (160,), (40,))
```

Train Model & Evaluate

```
model=DecisionTreeClassifier()
model.fit(X_train,Y_train)
```

```
DecisionTreeClassifier()
```

Evaluate

```
model.score(X_train, Y_train)
```

```
1.0
```

```
model.score(X_test, Y_test)
```

```
0.95
```

```
Y_pred = model.predict(X_test)
```

```
accuracy_score(Y_pred, Y_test)
```

```
0.95
```

```
print(classification_report(Y_pred, Y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      0.83      0.91        12
           2       1.00      1.00      1.00         9
           3       0.82      1.00      0.90         9
```