Tanmay Bhatkar TE INFT A Roll No=>9

Experiment 7
DSL
To implement Clustering

Laboratory Exercise
A. Procedure: (Mall_Customers Dataset)

Importing Libraries

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from mpl_toolkits.mplot3d import Axes3D
    %matplotlib inline
```

Reading Excel File

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

```
[ ] print("Number of customers we have data for-" , len(data))
```

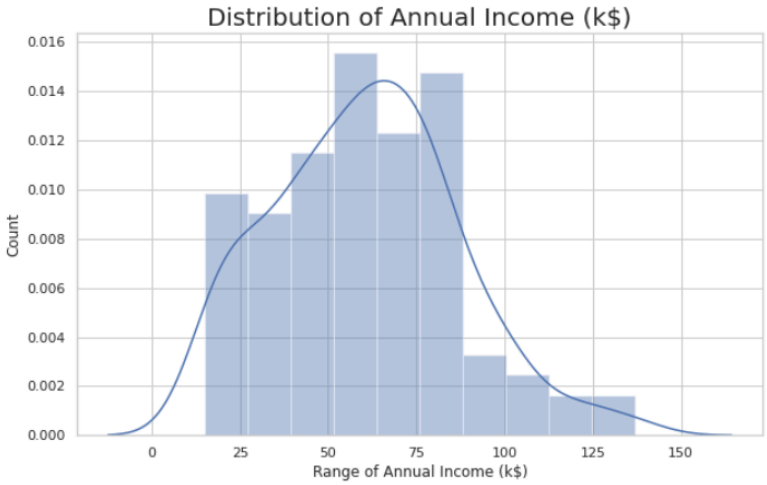
```
Number of customers we have data for- 200
```

```
[ ] data.head()
```

	CustomerID	Genre	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

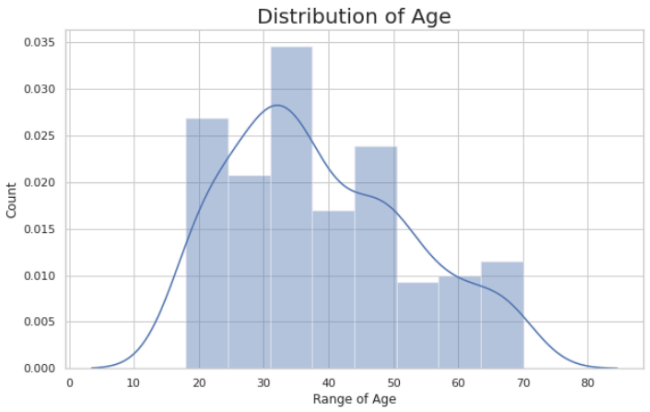
```
data.corr()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
CustomerID	1.000000	-0.026763	0.977548	0.013835
Age	-0.026763	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.327227	0.009903	1.000000



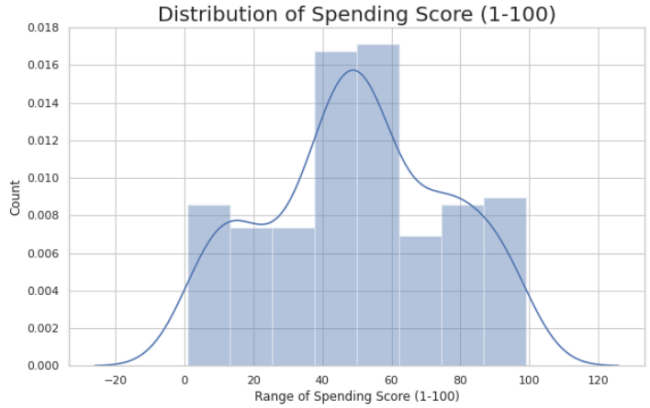

```
#Distribution of age

plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Age'])
plt.title('Distribution of Age', fontsize = 20)
plt.xlabel('Range of Age')
plt.ylabel('Count')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning
  warnings.warn(msg, FutureWarning)
Text(0, 0.5, 'Count')
```

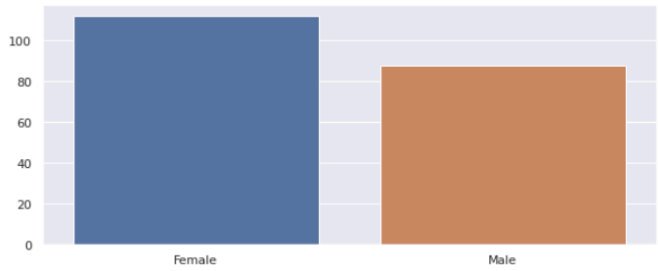


```
#Distribution of spending score

plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Spending Score (1-100)'])
plt.title('Distribution of Spending Score (1-100)', fontsize = 20)
plt.xlabel('Range of Spending Score (1-100)')
plt.ylabel('Count')
```

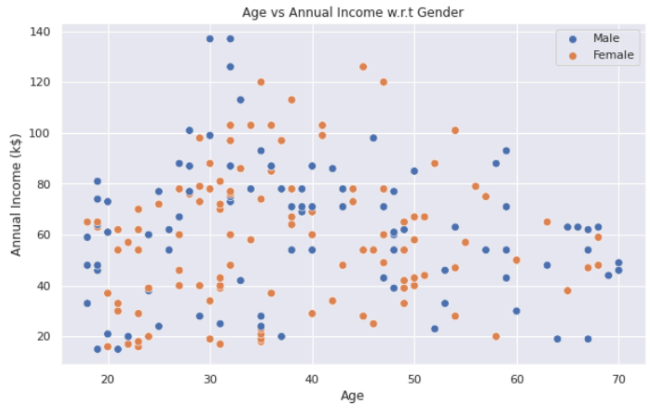
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning
  warnings.warn(msg, FutureWarning)
Text(0, 0.5, 'Count')
```



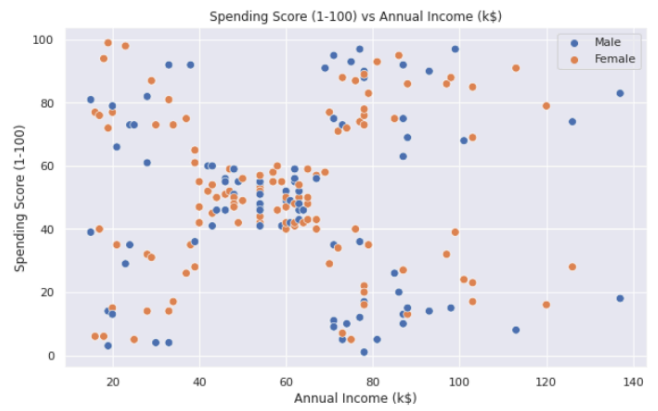

```
# Gender Analysis
genders = data.Genre.value_counts()
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.barplot(x=genders.index, y=genders.values)
plt.show()
```



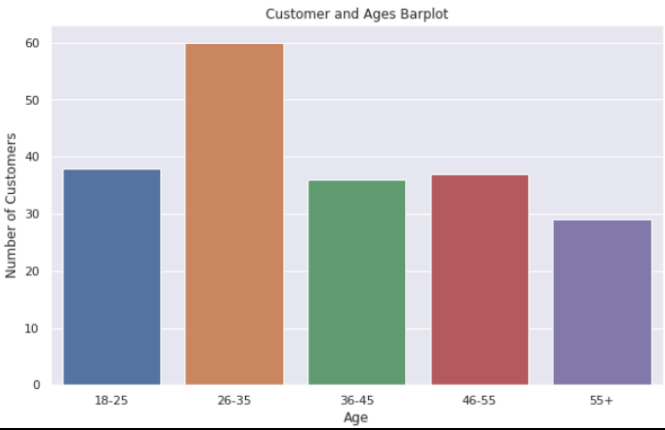

```
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Age' , y = 'Annual Income (k$)' , hue="Genre",data = data  ,s = 60 )
plt.xlabel('Age'), plt.ylabel('Annual Income (k$)')
plt.title('Age vs Annual Income w.r.t Gender')
plt.legend()
plt.show()
```




```
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',
                hue="Genre",data = data  ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.legend()
plt.show()
```




```
age18_25 = data.Age[(data.Age <= 25) & (data.Age >= 18)]
age26_35 = data.Age[(data.Age <= 35) & (data.Age >= 26)]
age36_45 = data.Age[(data.Age <= 45) & (data.Age >= 36)]
age46_55 = data.Age[(data.Age <= 55) & (data.Age >= 46)]
age55above = data.Age[data.Age >= 56]

x = ["18-25","26-35","36-45","46-55","55+"]
y = [len(age18_25.values),len(age26_35.values),len(age36_45.values),
     len(age46_55.values),len(age55above.values)]

plt.figure(figsize=(10,6))
sns.barplot(x=x, y=y)
plt.title("Customer and Ages Barplot")
plt.xlabel("Age")
plt.ylabel("Number of Customers")
plt.show()
```



Annual Income (1000 USD) BUckets

```
[ ] ai0_30 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 0) & (data["Annual Income (k$)"] <= 30)]
    ai31_60 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 31) & (data["Annual Income (k$)"] <= 60)]
    ai61_90 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 61) & (data["Annual Income (k$)"] <= 90)]
    ai91_120 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 91) & (data["Annual Income (k$)"] <= 120)]
    ai121_150 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 121) & (data["Annual Income (k$)"] <= 150)]

    income_x = ["$ 0 - 30,000", "$ 30,001 - 60,000", "$ 60,001 - 90,000", "$ 90,001 - 120,000", "$ 120,001 - 150,000"]
    income_y = [len(ai0_30.values), len(ai31_60.values), len(ai61_90.values), len(ai91_120.values), len(ai121_150.values)]

    plt.figure(figsize=(15,6))
    sns.barplot(x=income_x, y=income_y, palette="nipy_spectral_r")
    plt.title("Annual Incomes")
    plt.xlabel("Income")
    plt.ylabel("Number of Customer")
    plt.show()
```

```
[ ] data
```

```
[ ] #Fitting the input data

    km1.fit(X)
```

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

```
[ ] KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
           n_clusters=5, n_init=10,random_state=None, tol=0.0001, verbose=0)
```

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

```
[ ] #predicting the labels of the input data

    y=km1.predict(X)
```

```
[ ] #adding the labels to a column named label

    df1["label"] = y
```

```
[ ] #Taking 5 clusters

    km1=KMeans(n_clusters=5)
```

```
X
```

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
...	...	...
195	120	79
196	126	28
197	126	74
198	137	18
199	137	83

200 rows × 2 columns

```
#The new dataframe with the clustering done

df1.head()
```

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

```
#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','dodgerblue','red'], legend=
               'full',data = df1  ,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()
```


```
cust1=df1[df1["label"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("-----------------------------------------")
cust2=df1[df1["label"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("-----------------------------------------")
cust3=df1[df1["label"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("-----------------------------------------")
cust4=df1[df1["label"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
print("-----------------------------------------")
cust5=df1[df1["label"]==4]
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)
print("-----------------------------------------")
```

```
Number of customer in 1st group= 22
They are - [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 46]
-----------------------------------------
Number of customer in 2nd group= 23
They are - [ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45]
-----------------------------------------
Number of customer in 3rd group= 81
They are - [ 44  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
 118 119 120 121 122 123 127 133 143]
-----------------------------------------
Number of customer in 4th group= 39
They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158
 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194
 196 198 200]
-----------------------------------------
Number of customer in 5th group= 35
They are - [125 129 131 135 137 139 141 145 147 149 151 153 155 157 159 161 163 165
 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199]
-----------------------------------------
```

```
#Taking the features

X2=df2[["Age","Annual Income (k$)","Spending Score (1-100)"]]
```

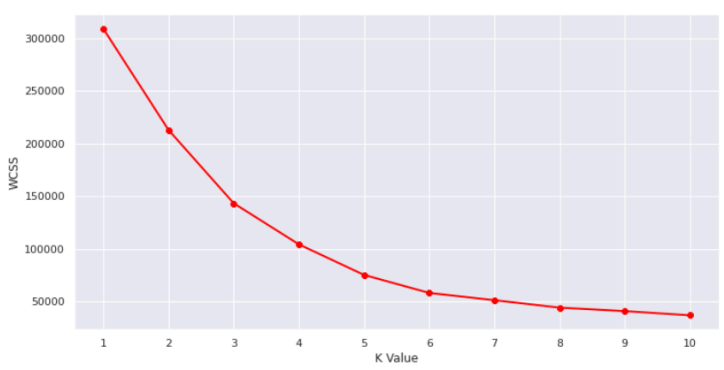
```
#Taking the features

X2=df2[["Age","Annual Income (k$)","Spending Score (1-100)"]]
```

```
#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k.


wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X2)
    wcss.append(kmeans.inertia_)
```

```
#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k.


wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X2)
    wcss.append(kmeans.inertia_)
```

```
plt.figure(figsize=(12,6))

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()
```

```
plt.figure(figsize=(12,6))

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()
```

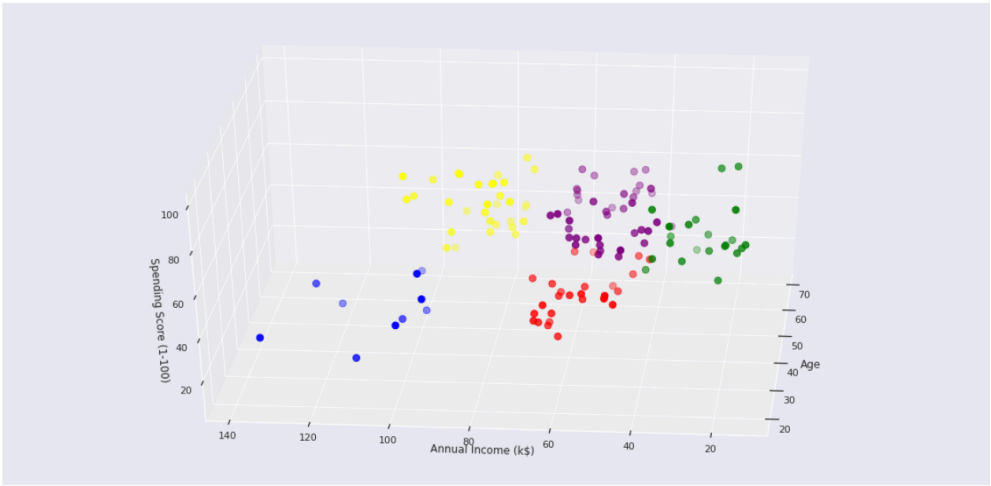



```
[ ]  #We choose the k for which WSS starts to diminish

     km2 = KMeans(n_clusters=5)
     y2 = km.fit_predict(X2)
     df2["label"] = y2
```

```
df2.head()
```

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

```
#3D Plot as we did the clustering on the basis of 3 input features

fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df2.Age[df2.label == 0], df2["Annual Income (k$)"][df2.label == 0], df2["Spending Score (1-100)"][df2.label == 0], c='purple', s=60)
ax.scatter(df2.Age[df2.label == 1], df2["Annual Income (k$)"][df2.label == 1], df2["Spending Score (1-100)"][df2.label == 1], c='red', s=60)
ax.scatter(df2.Age[df2.label == 2], df2["Annual Income (k$)"][df2.label == 2], df2["Spending Score (1-100)"][df2.label == 2], c='blue', s=60)
ax.scatter(df2.Age[df2.label == 3], df2["Annual Income (k$)"][df2.label == 3], df2["Spending Score (1-100)"][df2.label == 3], c='green', s=60)
ax.scatter(df2.Age[df2.label == 4], df2["Annual Income (k$)"][df2.label == 4], df2["Spending Score (1-100)"][df2.label == 4], c='yellow', s=60)
ax.view_init(35, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```




```
cust1=df2[df2["label"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("-----------------------------------------")
cust2=df2[df2["label"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("-----------------------------------------")
cust3=df2[df2["label"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("-----------------------------------------")
cust4=df2[df2["label"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
print("-----------------------------------------")
cust5=df2[df2["label"]==4]
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)
print("-----------------------------------------")
```

```
Number of customer in 1st group= 26
They are - [ 44  52  53  59  62  66  69  70  76  79  82  85  88  89  92  96  98 100
 101 104 106 112 114 115 116 121]
-----------------------------------------
Number of customer in 2nd group= 10
They are - [181 183 185 187 189 191 193 195 197 199]
-----------------------------------------
Number of customer in 3rd group= 41
They are - [ 41  47  51  54  55  57  58  60  61  63  64  65  67  68  71  72  73  74
  75  77  80  81  83  84  86  87  90  91  93  97 102 103 105 107 108 109
 110 111 117 118 120]
-----------------------------------------
Number of customer in 4th group= 22
They are - [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 46]
-----------------------------------------
Number of customer in 5th group= 32
They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158
 160 162 164 166 168 170 172 174 176 178 180 182 184 186]
-----------------------------------------
```

8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

● Types of clustering.

There are several types of clustering methods, each with different strategies for defining and identifying clusters. The main types are:

1. Partitioning Clustering

Algorithm: K-Means, K-Medoids

How it works: Divides data into K non-overlapping subsets (clusters) without any hierarchy.

Best for: When the number of clusters is known in advance.

Limitation: Sensitive to outliers, and K needs to be specified.

2. Hierarchical Clustering

Algorithm: Agglomerative (bottom-up), Divisive (top-down)

How it works: Creates a tree-like structure (dendrogram) by successively merging or splitting clusters.

Best for: When you want to understand the data hierarchy.

Limitation: Computationally expensive for large datasets.

3. Density-Based Clustering

Algorithm: DBSCAN, OPTICS

How it works: Groups together points that are closely packed, and marks points in low-density regions as outliers.

Best for: Discovering clusters of arbitrary shapes and handling noise.

Limitation: Poor performance with varying densities.

4. Model-Based Clustering

Algorithm: Gaussian Mixture Models (GMM)

How it works: Assumes the data is generated from a mixture of several Gaussian distributions.

Best for: When data is probabilistic or soft-clustering is needed.

Limitation: Assumes distribution form; may overfit if not regularized.

5. Grid-Based Clustering

Algorithm: STING, CLIQUE

How it works: Divides the data space into a grid and performs clustering on the grid structure.

Best for: Large datasets with fast processing needs.

Limitation: Not ideal for discovering clusters of irregular shapes.