gmentation-with-k-means-clustering

April 29, 2024

[26]:
```
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
warnings.filterwarnings("ignore")
```

0.1 Step 1: Data Exploration and Preprocessing

- [x] Load your customer dataset.
- [x] Check for missing values.
- [x] Check data types.
- [x] Handle missing values (impute or drop).
- [x] Check Duplicates
- [x] Checking Outliers

[27]:
```
df=pd.read_csv("Mall_Customers.csv")
```

[28]:
```
df.head(5)
```

[28]:

	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

[29]:
```
#shape of the data
df.shape
```

[29]:
```
(200, 5)
```

[30]:
```
# Checking Datatypes
df.dtypes
```

[30]:
```
CustomerID                int64
Gender                   object
Age                       int64
Annual Income (k$)        int64
Spending Score (1-100)    int64
dtype: object
```

[31]:
```
#checking description of data
df.describe()
```

[31]:

	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

Checking for missing values

[32]:
```
df.isna().sum()
```

[32]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

[33]:
```
df.isna().sum().sum()
```

[33]:
```
0
```

No NUll Values

[ ]:

0.1.1 Checking Duplicated

[34]:
```
df.duplicated().sum()
```

[34]:
```
0
```

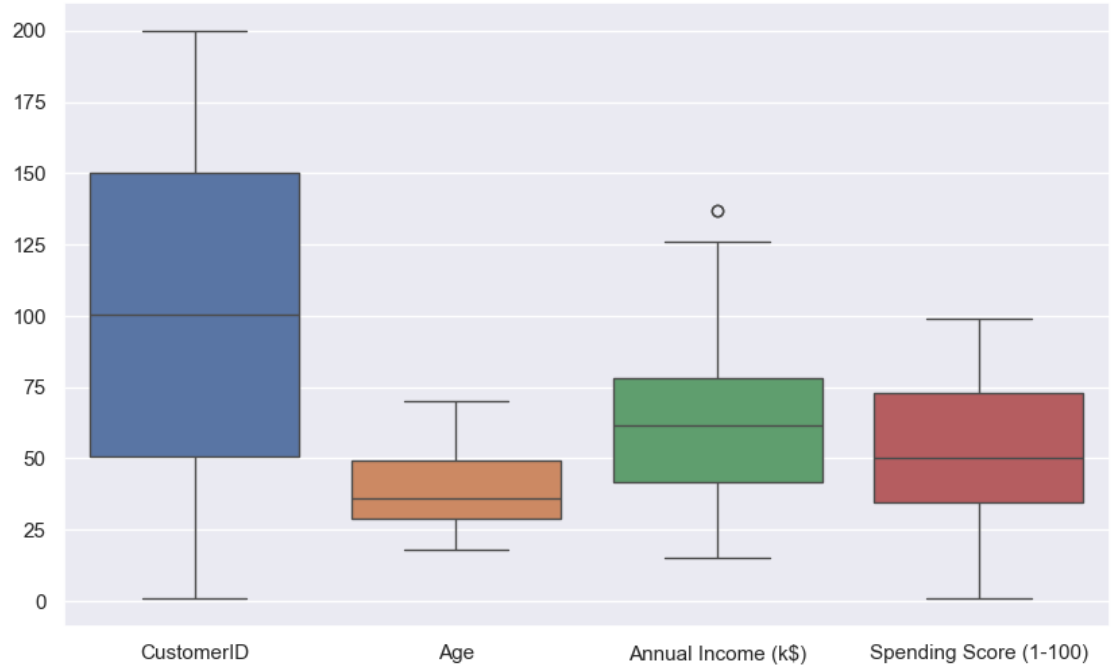
No Duplicates

[ ]:

0.1.2 Checking Outliers

[35]:
```
plt.figure(figsize=(10,6))
sns.boxplot(data=df)
```

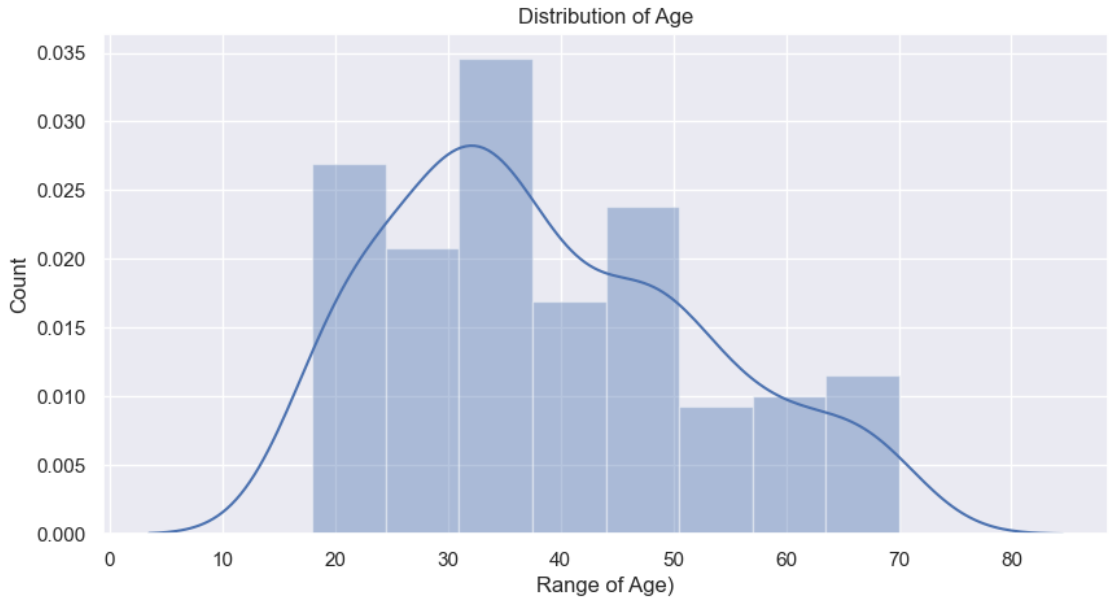
[35]: <Axes: >



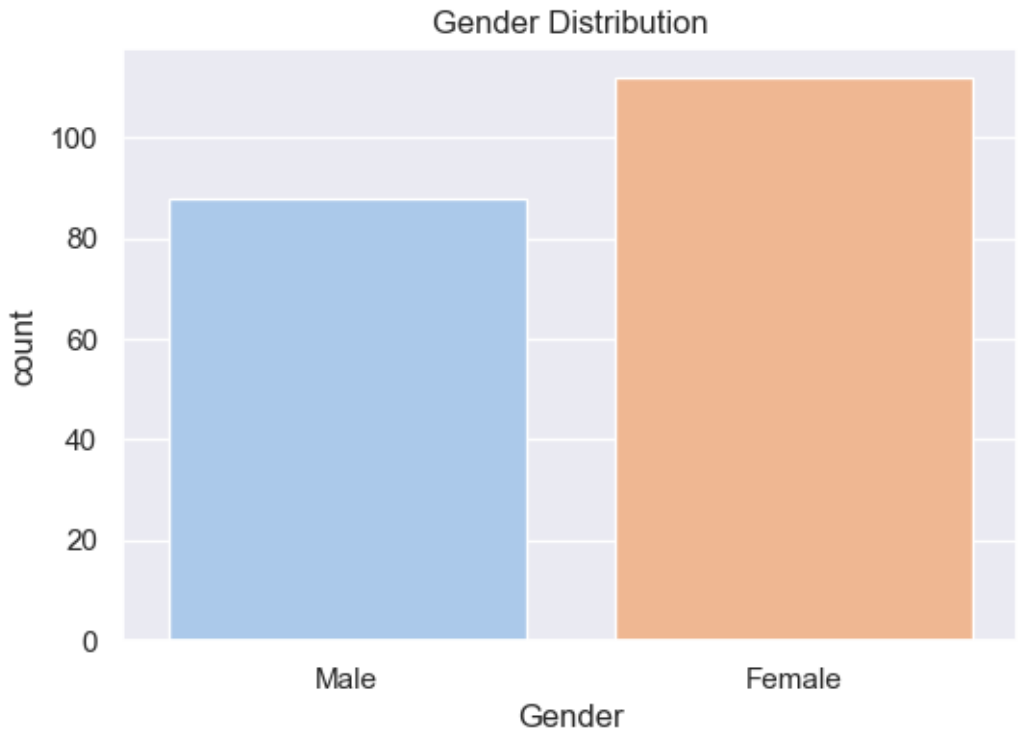
0.2 Step 2: Data Visualization

[36]:
```
#Distribution of Annnual Income
plt.figure(figsize=(10, 5))
sns.set(style = 'darkgrid')
sns.distplot(df['Annual Income (k$)'])
plt.title('Distribution of Annual Income (k$)')
plt.xlabel('Range of Annual Income (k$)')
plt.ylabel('Count')
plt.show()
```

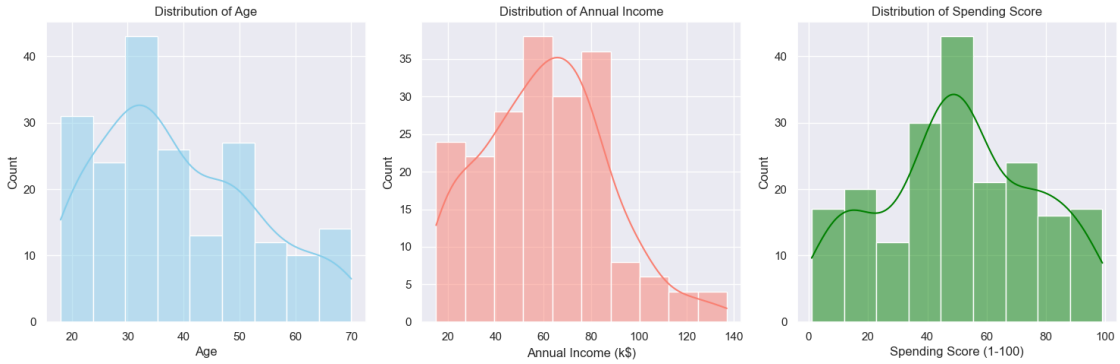
[37]:
```
#Distribution of Annnual Income
plt.figure(figsize=(10, 5))
sns.set(style = 'darkgrid')
sns.distplot(df['Age'])
plt.title('Distribution of Age')
plt.xlabel('Range of Age)')
plt.ylabel('Count')
plt.show()
```



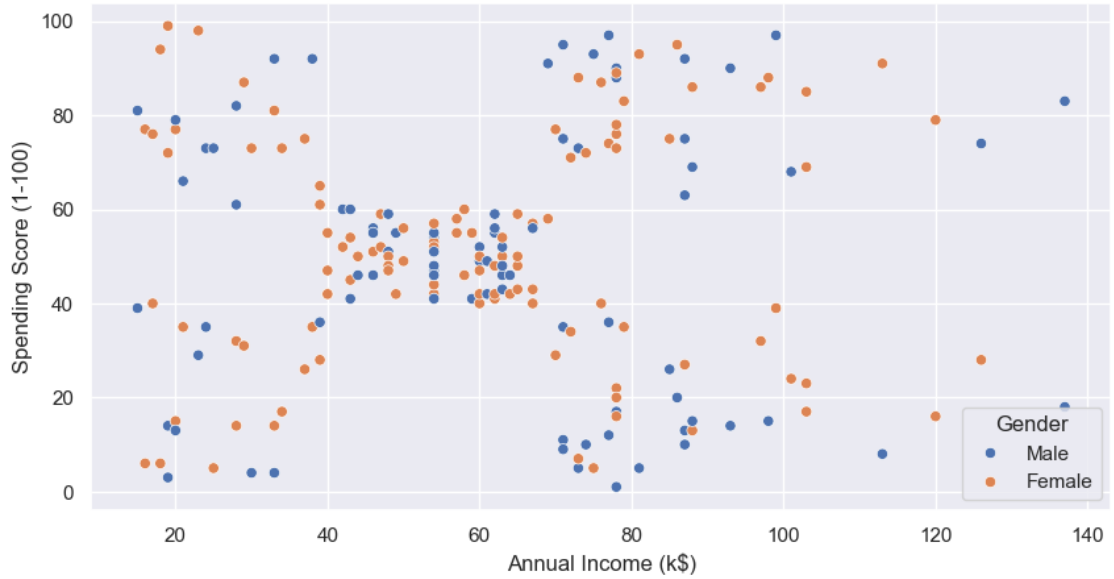
[38]:
```
plt.figure(figsize=(6, 4))
sns.countplot(x='Gender', data=df, palette='pastel')
plt.title('Gender Distribution')
plt.show()
```



[39]:
```
# Histograms for numerical features
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.histplot(df['Age'], kde=True, color='skyblue')
plt.title('Distribution of Age')

plt.subplot(1, 3, 2)
sns.histplot(df['Annual Income (k$)'], kde=True, color='salmon')
plt.title('Distribution of Annual Income')

plt.subplot(1, 3, 3)
sns.histplot(df['Spending Score (1-100)'], kde=True, color='green')
plt.title('Distribution of Spending Score')

plt.tight_layout()
plt.show()
```

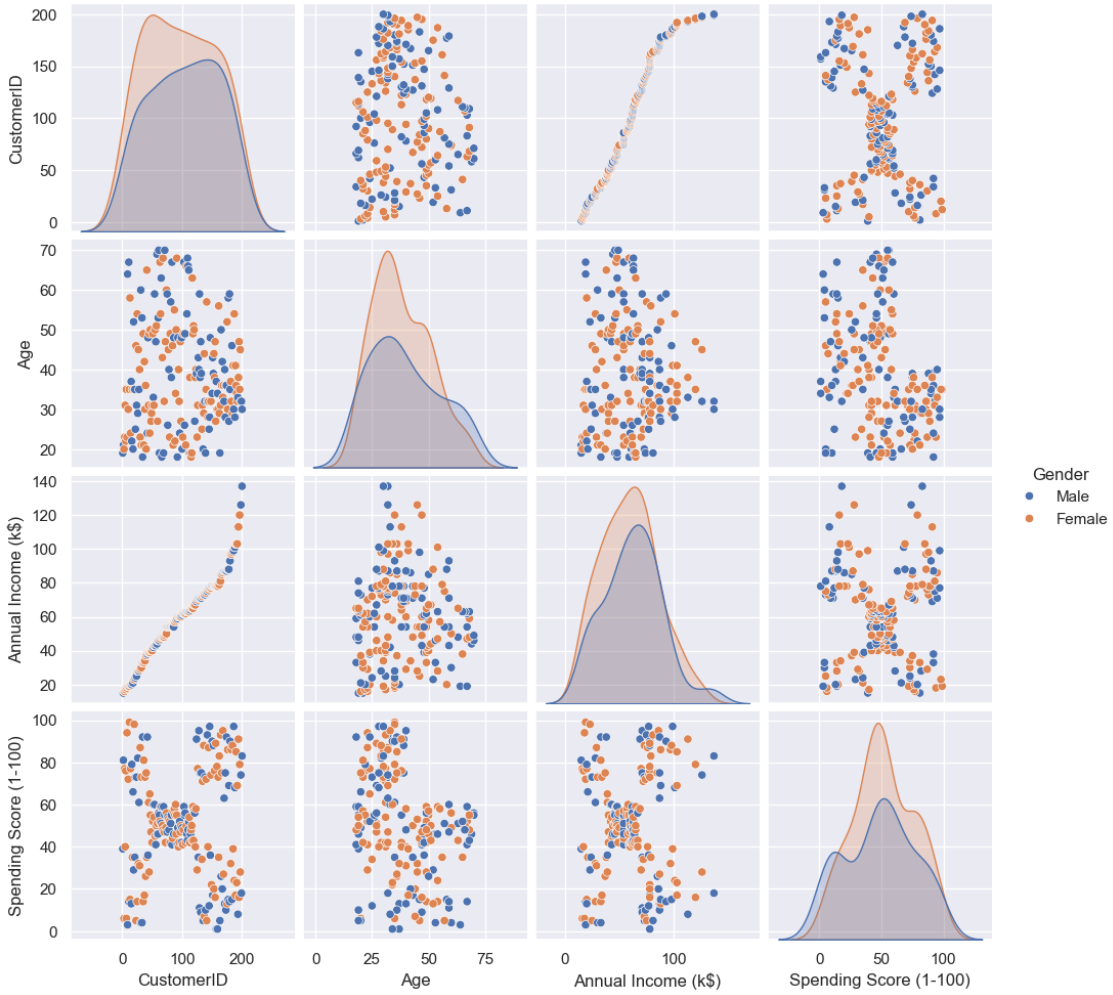


[40]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(data=df,x="Annual Income (k$)",y="Spending Score␣
↪(1-100)",hue="Gender")
plt.show()
```



[41]:
```
plt.figure(figsize=(10,15))
sns.pairplot(data=df,hue="Gender")
plt.show()
```

```
<Figure size 1000x1500 with 0 Axes>
```



0.3 Step 3: Preprocess the data by scaling the features

[42]:
```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.iloc[:, 2:])
```

[43]:
```
scaler
```

[43]: StandardScaler()

0.4 Step 4: Select the relevant features

[44]:
```
selected_features = scaled_features
```

[45]:
```
selected_features
```

[45]:
```
array([[-1.42456879, -1.73899919, -0.43480148],
       [-1.28103541, -1.73899919,  1.19570407],
       [-1.3528021 , -1.70082976, -1.71591298],
       [-1.13750203, -1.70082976,  1.04041783],
       [-0.56336851, -1.66266033, -0.39597992],
       [-1.20926872, -1.66266033,  1.00159627],
       [-0.27630176, -1.62449091, -1.71591298],
       [-1.13750203, -1.62449091,  1.70038436],
       [ 1.80493225, -1.58632148, -1.83237767],
       [-0.6351352 , -1.58632148,  0.84631002],
       [ 2.02023231, -1.58632148, -1.4053405 ],
       [-0.27630176, -1.58632148,  1.89449216],
       [ 1.37433211, -1.54815205, -1.36651894],
       [-1.06573534, -1.54815205,  1.04041783],
       [-0.13276838, -1.54815205, -1.44416206],
       [-1.20926872, -1.54815205,  1.11806095],
       [-0.27630176, -1.50998262, -0.59008772],
       [-1.3528021 , -1.50998262,  0.61338066],
       [ 0.94373197, -1.43364376, -0.82301709],
       [-0.27630176, -1.43364376,  1.8556706 ],
       [-0.27630176, -1.39547433, -0.59008772],
       [-0.99396865, -1.39547433,  0.88513158],
       [ 0.51313183, -1.3573049 , -1.75473454],
       [-0.56336851, -1.3573049 ,  0.88513158],
       [ 1.08726535, -1.24279661, -1.4053405 ],
       [-0.70690189, -1.24279661,  1.23452563],
       [ 0.44136514, -1.24279661, -0.7065524 ],
       [-0.27630176, -1.24279661,  0.41927286],
       [ 0.08253169, -1.20462718, -0.74537397],
       [-1.13750203, -1.20462718,  1.42863343],
       [ 1.51786549, -1.16645776, -1.7935561 ],
       [-1.28103541, -1.16645776,  0.88513158],
       [ 1.01549866, -1.05194947, -1.7935561 ],
       [-1.49633548, -1.05194947,  1.62274124],
       [ 0.7284319 , -1.05194947, -1.4053405 ],
       [-1.28103541, -1.05194947,  1.19570407],
       [ 0.22606507, -1.01378004, -1.28887582],
       [-0.6351352 , -1.01378004,  0.88513158],
       [-0.20453507, -0.89927175, -0.93948177],
       [-1.3528021 , -0.89927175,  0.96277471],
       [ 1.87669894, -0.86110232, -0.59008772],
```

```
[-1.06573534, -0.86110232,  1.62274124],
[ 0.65666521, -0.82293289, -0.55126616],
[-0.56336851, -0.82293289,  0.41927286],
[ 0.7284319 , -0.82293289, -0.86183865],
[-1.06573534, -0.82293289,  0.5745591 ],
[ 0.80019859, -0.78476346,  0.18634349],
[-0.85043527, -0.78476346, -0.12422899],
[-0.70690189, -0.78476346, -0.3183368 ],
[-0.56336851, -0.78476346, -0.3183368 ],
[ 0.7284319 , -0.70842461,  0.06987881],
[-0.41983513, -0.70842461,  0.38045129],
[-0.56336851, -0.67025518,  0.14752193],
[ 1.4460988 , -0.67025518,  0.38045129],
[ 0.80019859, -0.67025518, -0.20187212],
[ 0.58489852, -0.67025518, -0.35715836],
[ 0.87196528, -0.63208575, -0.00776431],
[ 2.16376569, -0.63208575, -0.16305055],
[-0.85043527, -0.55574689,  0.03105725],
[ 1.01549866, -0.55574689, -0.16305055],
[ 2.23553238, -0.55574689,  0.22516505],
[-1.42456879, -0.55574689,  0.18634349],
[ 2.02023231, -0.51757746,  0.06987881],
[ 1.08726535, -0.51757746,  0.34162973],
[ 1.73316556, -0.47940803,  0.03105725],
[-1.49633548, -0.47940803,  0.34162973],
[ 0.29783176, -0.47940803, -0.00776431],
[ 2.091999  , -0.47940803, -0.08540743],
[-1.42456879, -0.47940803,  0.34162973],
[-0.49160182, -0.47940803, -0.12422899],
[ 2.23553238, -0.4412386 ,  0.18634349],
[ 0.58489852, -0.4412386 , -0.3183368 ],
[ 1.51786549, -0.40306917, -0.04658587],
[ 1.51786549, -0.40306917,  0.22516505],
[ 1.4460988 , -0.25039146, -0.12422899],
[-0.92220196, -0.25039146,  0.14752193],
[ 0.44136514, -0.25039146,  0.10870037],
[ 0.08253169, -0.25039146, -0.08540743],
[-1.13750203, -0.25039146,  0.06987881],
[ 0.7284319 , -0.25039146, -0.3183368 ],
[ 1.30256542, -0.25039146,  0.03105725],
[-0.06100169, -0.25039146,  0.18634349],
[ 2.02023231, -0.25039146, -0.35715836],
[ 0.51313183, -0.25039146, -0.24069368],
[-1.28103541, -0.25039146,  0.26398661],
[ 0.65666521, -0.25039146, -0.16305055],
[ 1.15903204, -0.13588317,  0.30280817],
[-1.20926872, -0.13588317,  0.18634349],
```

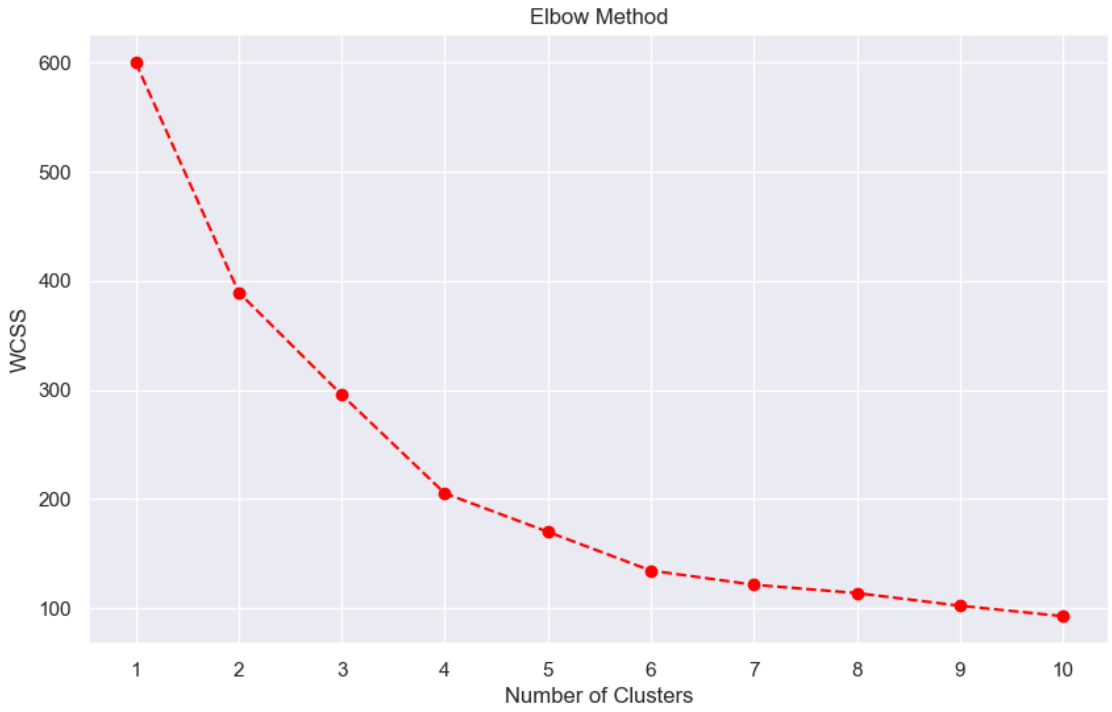
```
[-0.34806844, -0.09771374,  0.38045129],
[ 0.80019859, -0.09771374, -0.16305055],
[ 2.091999  , -0.05954431,  0.18634349],
[-1.49633548, -0.05954431, -0.35715836],
[ 0.65666521, -0.02137488, -0.04658587],
[ 0.08253169, -0.02137488, -0.39597992],
[-0.49160182, -0.02137488, -0.3183368 ],
[-1.06573534, -0.02137488,  0.06987881],
[ 0.58489852, -0.02137488, -0.12422899],
[-0.85043527, -0.02137488, -0.00776431],
[ 0.65666521,  0.01679455, -0.3183368 ],
[-1.3528021 ,  0.01679455, -0.04658587],
[-1.13750203,  0.05496398, -0.35715836],
[ 0.7284319 ,  0.05496398, -0.08540743],
[ 2.02023231,  0.05496398,  0.34162973],
[-0.92220196,  0.05496398,  0.18634349],
[ 0.7284319 ,  0.05496398,  0.22516505],
[-1.28103541,  0.05496398, -0.3183368 ],
[ 1.94846562,  0.09313341, -0.00776431],
[ 1.08726535,  0.09313341, -0.16305055],
[ 2.091999  ,  0.09313341, -0.27951524],
[ 1.94846562,  0.09313341, -0.08540743],
[ 1.87669894,  0.09313341,  0.06987881],
[-1.42456879,  0.09313341,  0.14752193],
[-0.06100169,  0.13130284, -0.3183368 ],
[-1.42456879,  0.13130284, -0.16305055],
[-1.49633548,  0.16947227, -0.08540743],
[-1.42456879,  0.16947227, -0.00776431],
[ 1.73316556,  0.16947227, -0.27951524],
[ 0.7284319 ,  0.16947227,  0.34162973],
[ 0.87196528,  0.24581112, -0.27951524],
[ 0.80019859,  0.24581112,  0.26398661],
[-0.85043527,  0.24581112,  0.22516505],
[-0.06100169,  0.24581112, -0.39597992],
[ 0.08253169,  0.32214998,  0.30280817],
[ 0.010765  ,  0.32214998,  1.58391968],
[-1.13750203,  0.36031941, -0.82301709],
[-0.56336851,  0.36031941,  1.04041783],
[ 0.29783176,  0.39848884, -0.59008772],
[ 0.08253169,  0.39848884,  1.73920592],
[ 1.4460988 ,  0.39848884, -1.52180518],
[-0.06100169,  0.39848884,  0.96277471],
[ 0.58489852,  0.39848884, -1.5994483 ],
[ 0.010765  ,  0.39848884,  0.96277471],
[-0.99396865,  0.43665827, -0.62890928],
[-0.56336851,  0.43665827,  0.80748846],
[-1.3528021 ,  0.4748277 , -1.75473454],
```

```
[-0.70690189,  0.4748277 ,  1.46745499],
[ 0.36959845,  0.4748277 , -1.67709142],
[-0.49160182,  0.4748277 ,  0.88513158],
[-1.42456879,  0.51299713, -1.56062674],
[-0.27630176,  0.51299713,  0.84631002],
[ 1.30256542,  0.55116656, -1.75473454],
[-0.49160182,  0.55116656,  1.6615628 ],
[-0.77866858,  0.58933599, -0.39597992],
[-0.49160182,  0.58933599,  1.42863343],
[-0.99396865,  0.62750542, -1.48298362],
[-0.77866858,  0.62750542,  1.81684904],
[ 0.65666521,  0.62750542, -0.55126616],
[-0.49160182,  0.62750542,  0.92395314],
[-0.34806844,  0.66567484, -1.09476801],
[-0.34806844,  0.66567484,  1.54509812],
[ 0.29783176,  0.66567484, -1.28887582],
[ 0.010765  ,  0.66567484,  1.46745499],
[ 0.36959845,  0.66567484, -1.17241113],
[-0.06100169,  0.66567484,  1.00159627],
[ 0.58489852,  0.66567484, -1.32769738],
[-0.85043527,  0.66567484,  1.50627656],
[-0.13276838,  0.66567484, -1.91002079],
[-0.6351352 ,  0.66567484,  1.07923939],
[-0.34806844,  0.66567484, -1.91002079],
[-0.6351352 ,  0.66567484,  0.88513158],
[ 1.23079873,  0.70384427, -0.59008772],
[-0.70690189,  0.70384427,  1.27334719],
[-1.42456879,  0.78018313, -1.75473454],
[-0.56336851,  0.78018313,  1.6615628 ],
[ 0.80019859,  0.93286085, -0.93948177],
[-0.20453507,  0.93286085,  0.96277471],
[ 0.22606507,  0.97103028, -1.17241113],
[-0.41983513,  0.97103028,  1.73920592],
[-0.20453507,  1.00919971, -0.90066021],
[-0.49160182,  1.00919971,  0.49691598],
[ 0.08253169,  1.00919971, -1.44416206],
[-0.77866858,  1.00919971,  0.96277471],
[-0.20453507,  1.00919971, -1.56062674],
[-0.20453507,  1.00919971,  1.62274124],
[ 0.94373197,  1.04736914, -1.44416206],
[-0.6351352 ,  1.04736914,  1.38981187],
[ 1.37433211,  1.04736914, -1.36651894],
[-0.85043527,  1.04736914,  0.72984534],
[ 1.4460988 ,  1.23821628, -1.4053405 ],
[-0.27630176,  1.23821628,  1.54509812],
[-0.13276838,  1.390894  , -0.7065524 ],
[-0.49160182,  1.390894  ,  1.38981187],
```

```
[ 0.51313183,  1.42906343, -1.36651894],
[-0.70690189,  1.42906343,  1.46745499],
[ 0.15429838,  1.46723286, -0.43480148],
[-0.6351352 ,  1.46723286,  1.81684904],
[ 1.08726535,  1.54357172, -1.01712489],
[-0.77866858,  1.54357172,  0.69102378],
[ 0.15429838,  1.61991057, -1.28887582],
[-0.20453507,  1.61991057,  1.35099031],
[-0.34806844,  1.61991057, -1.05594645],
[-0.49160182,  1.61991057,  0.72984534],
[-0.41983513,  2.00160487, -1.63826986],
[-0.06100169,  2.00160487,  1.58391968],
[ 0.58489852,  2.26879087, -1.32769738],
[-0.27630176,  2.26879087,  1.11806095],
[ 0.44136514,  2.49780745, -0.86183865],
[-0.49160182,  2.49780745,  0.92395314],
[-0.49160182,  2.91767117, -1.25005425],
[-0.6351352 ,  2.91767117,  1.27334719]])
```

0.5 Step 5:optimal number of clusters using techniques like the elbow method and then apply k-mean clustering algorithm

[47]:
```
from sklearn.cluster import KMeans

wcss_list = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(selected_features)
    wcss_list.append(kmeans.inertia_)

# Plot the elbow curve
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss_list, marker='o', linestyle='--',c="red")
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.xticks(np.arange(1, 11, 1))
plt.grid(True)
plt.show()
```



0.6 Step 6: Apply K-means clustering

[48]:
```
from sklearn.cluster import KMeans

num_clusters = 5
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
df['cluster'] = kmeans.fit_predict(selected_features)
```

[49]:
```
kmeans.labels_
```

[49]:
```
array([2, 2, 3, 2, 2, 2, 3, 2, 0, 2, 0, 2, 0, 2, 3, 2, 3, 2, 0, 2, 3, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 3, 2, 0, 2, 0, 2,
       0, 2, 0, 3, 3, 3, 0, 2, 3, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 3,
       0, 0, 3, 3, 0, 0, 0, 0, 0, 3, 0, 3, 3, 0, 0, 3, 0, 0, 3, 0, 0, 3,
       3, 0, 0, 3, 0, 3, 3, 3, 0, 3, 0, 3, 3, 0, 0, 3, 0, 3, 0, 0, 0, 0,
       0, 3, 3, 3, 3, 3, 0, 0, 0, 0, 3, 3, 1, 1, 3, 1, 4, 1, 4, 1, 4, 1,
       3, 1, 3, 1, 4, 1, 3, 1, 4, 1, 3, 1, 3, 1, 4, 1, 4, 1, 4, 1, 4, 1,
       4, 1, 4, 1, 4, 1, 4, 1, 3, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1,
       4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1,
       4, 1])
```

[50]:
```
df
```

[50]:
```
     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

     cluster
0          2
1          2
2          3
3          2
4          2
..       …
195        1
196        4
197        1
198        4
199        1

[200 rows x 6 columns]
```

0.7 Step 6: Visualize the resulting clusters

[51]:
```
#Scatterplot of the clusters
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score␣
 ↪(1-100)',hue="cluster",
                 palette=['green','orange','brown','dodgerblue','red'],␣
 ↪legend='full',data = df  ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```

[52]:
```
plt.figure(figsize=(10, 6))
colors = ['purple', 'red', 'blue', 'green', 'yellow']
for i in range(5):
    plt.scatter(df["Age"][df.cluster == i], df["Annual Income (k$)"][df.cluster
 ↪== i], c=colors[i], label=f'Cluster {i+1}', s=60)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
plt.title('Customer Segmentation')
plt.legend()
plt.show()
```

[55]:
```
from mpl_toolkits.mplot3d import Axes3D

# Plotting the 3D scatter plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot for each cluster
for i in range(5):
    ax.scatter(df[df['cluster'] == i]['Age'],
               df[df['cluster'] == i]['Annual Income (k$)'],
               df[df['cluster'] == i]['Spending Score (1-100)'],
               label=f'Cluster {i+1}')

ax.set_xlabel('Age')
ax.set_ylabel('Annual Income (k$)')
ax.set_zlabel('Spending Score (1-100)')
ax.set_title('Customer Segmentation')

plt.legend()
plt.show()
```

[56]:
```
for i in range(5):
    cust = df[df["cluster"]==i]
    print(f'Number of customers in {i+1}st group = {len(cust)}')
    print(f'They are - {cust["CustomerID"].values}')
    print("-----------------------------------------")
```

```
Number of customers in 1st group = 58
They are - [  9  11  13  19  23  25  27  29  31  33  35  37  41  43  45  47  51
54
  55  56  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  99 102 103 105 107 108 109 110 111
 117 118 119 120]
-----------------------------------------
Number of customers in 2st group = 40
They are - [123 124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154
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 customers in 3st group = 26
They are - [ 1  2  4  5  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42
44
 46 52]
-----------------------------------------
Number of customers in 4st group = 45
They are - [  3   7  15  17  21  39  48  49  50  53  59  62  66  69  70  76  78
79
  82  85  88  89  92  94  95  96  98 100 101 104 106 112 113 114 115 116
 121 122 125 133 135 139 143 145 163]
-----------------------------------------
Number of customers in 5st group = 31
They are - [127 129 131 137 141 147 149 151 153 155 157 159 161 165 167 169 171
173
 175 177 179 181 183 185 187 189 191 193 195 197 199]
-----------------------------------------
```

[ ]: