Task 2

August 17, 2023

1 Task 2 - Use Clustering Techniques for the any customer dataset using machine learning

2 Problem Statement

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

2.1 Importing required libraries

[70]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN, MeanShift,␣
↪Birch

from sklearn.preprocessing import StandardScaler

from scipy.cluster.hierarchy import dendrogram,linkage

from sklearn.mixture import GaussianMixture
```

2.2 Loading the dataset

[2]:
```
df = pd.read_csv("Mall_Customers.csv")
```

2.3 Getting to know about the data

[3]:
```
df.sample(5)
```

[3]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
102	103	Male	67	62	59
96	97	Female	47	60	47
15	16	Male	22	20	79
61	62	Male	19	46	55
85	86	Male	48	54	46

[4]:
```
df.shape
```

[4]:
```
(200, 5)
```

[5]:
```
df.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   Genre                   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
```

[6]:
```
df.describe()
```

[6]:

	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

2.4 Checking for null values

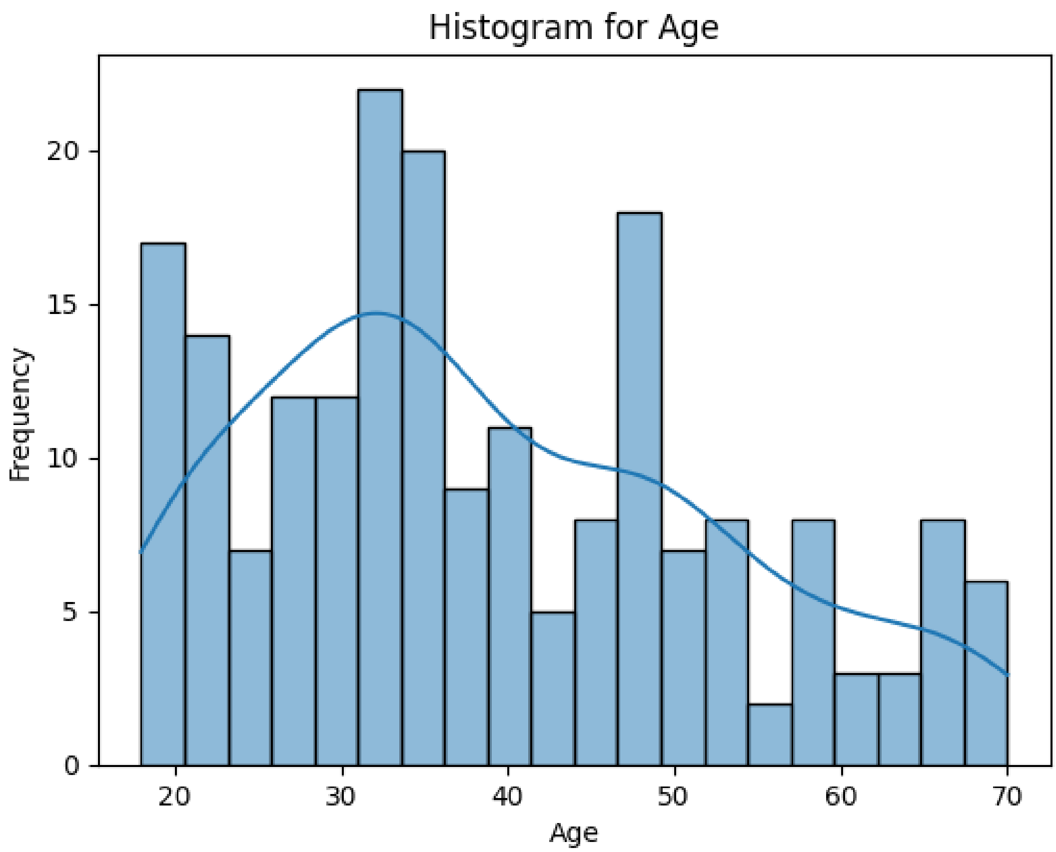
[7]:
```
df.isnull().sum()
```

[7]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

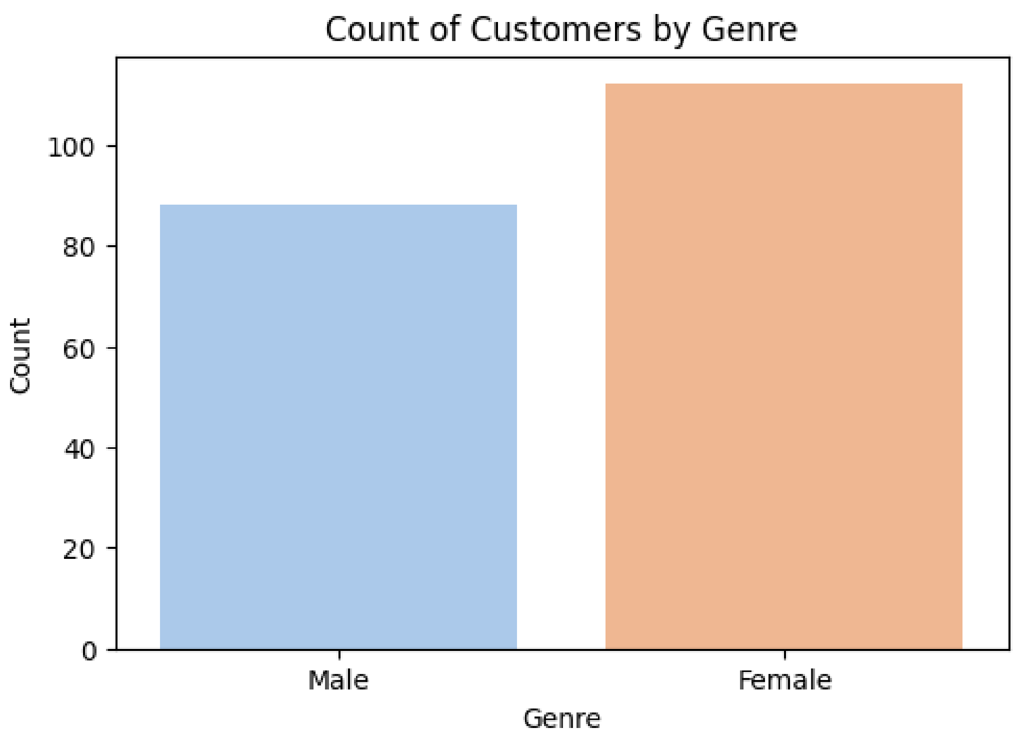
- The dataset doesn't have any missing values

2.5 Exploratory Data Analysis

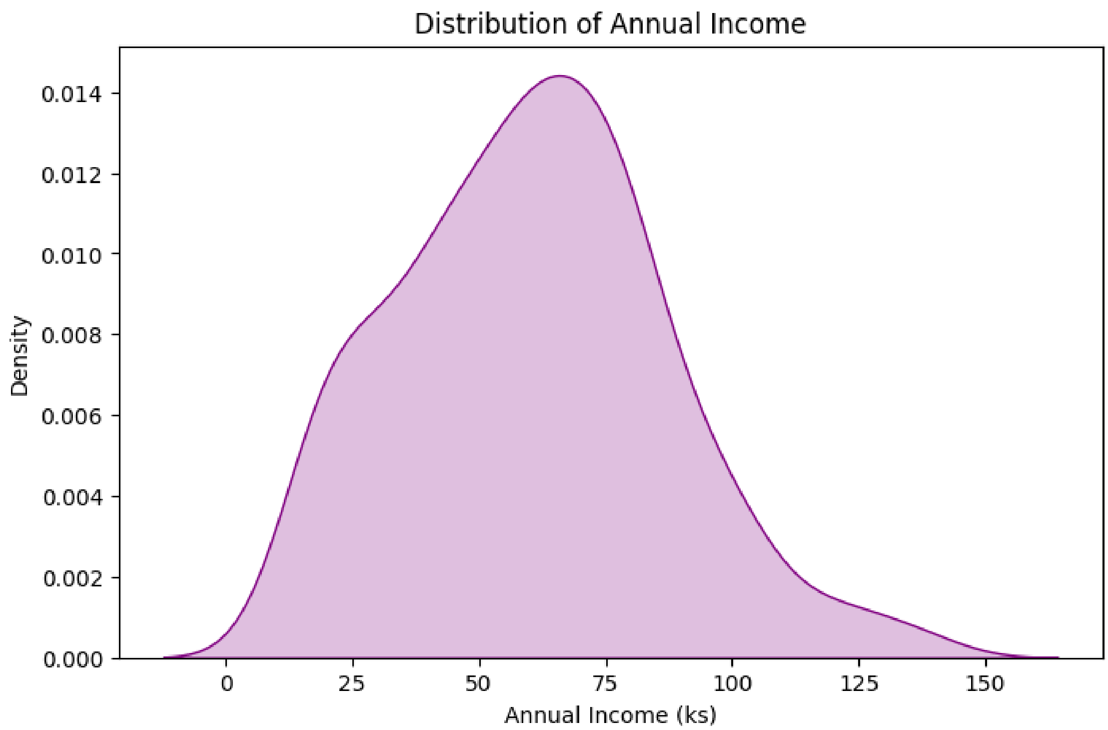
[8]:
```
sns.histplot(data = df['Age'], kde = True,bins = 20)
plt.title("Histogram for Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```



[9]:
```
plt.figure(figsize=(6, 4))
sns.countplot(x='Genre',data= df, palette='pastel')
plt.title('Count of Customers by Genre')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.show()
```



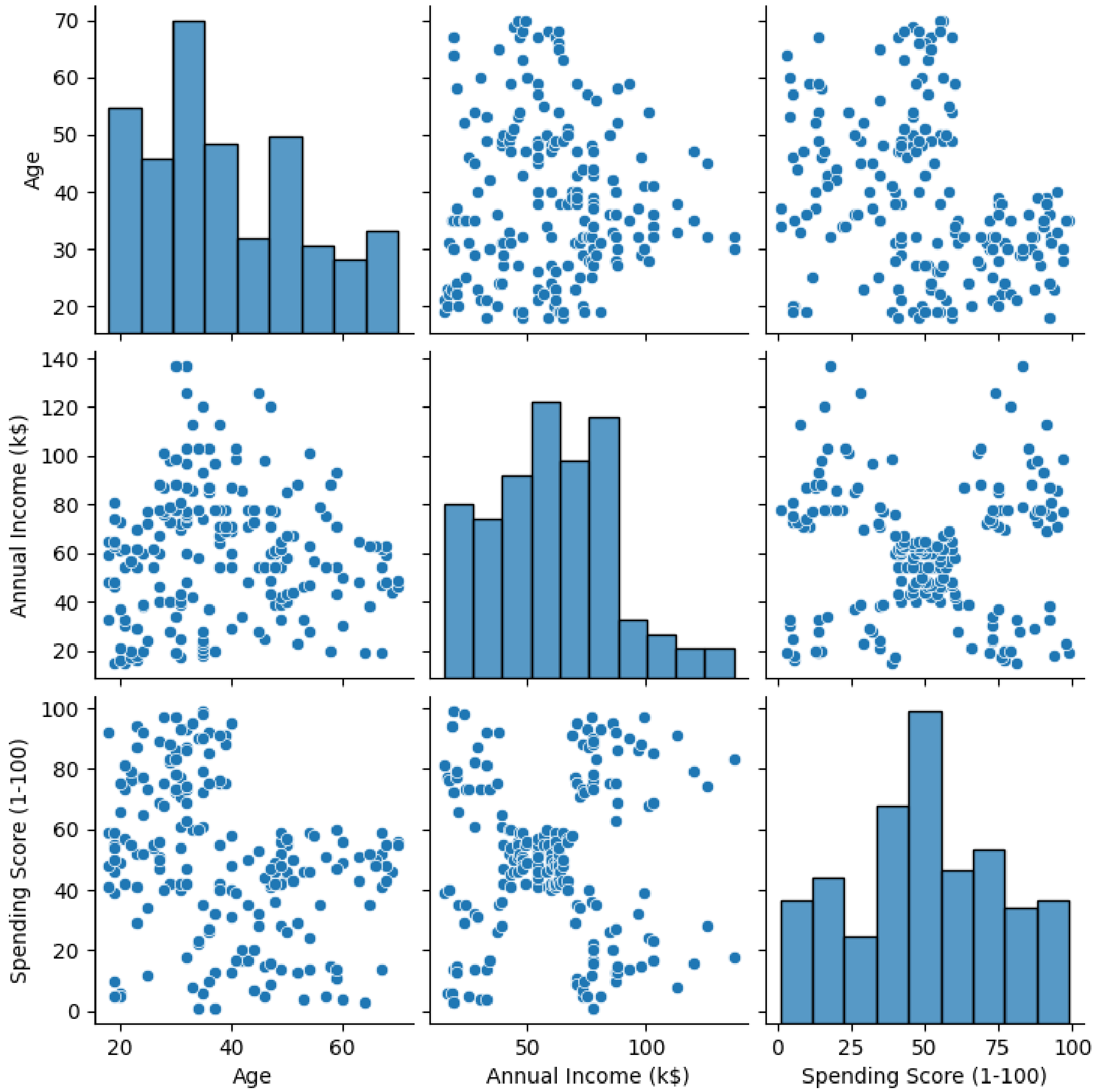
[10]:
```
plt.figure(figsize=(8, 5))
sns.kdeplot(df['Annual Income (k$)'], shade=True, color='purple')
plt.title('Distribution of Annual Income')
plt.xlabel('Annual Income (ks)')
plt.ylabel('Density')
plt.show()
```



[11]:
```
plt.figure(figsize=(8, 6))
sns.boxplot (x='Genre', y='Spending Score (1-100)', data= df, palette="pastel")
plt.xlabel('Genre')
plt.title('Box plot for Spending Score across Genre')
plt.ylabel('Spending Score (1-100)')
plt.show()
```

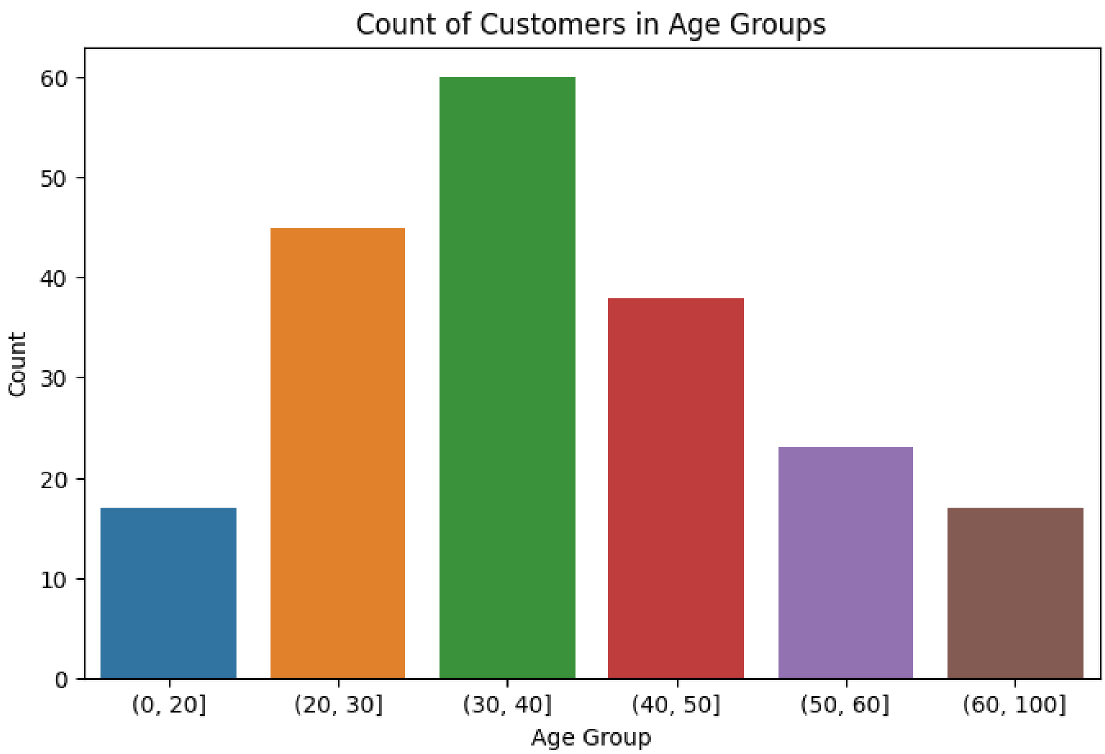
2.5.1 'Pairplot for Age, Annual Income, and Spending Score'

[12]:
```
sns.pairplot(df[['Age', 'Annual Income (k$)', 'Spending Score␣
↪(1-100)']],diag_kind='hist')
plt.show()
```



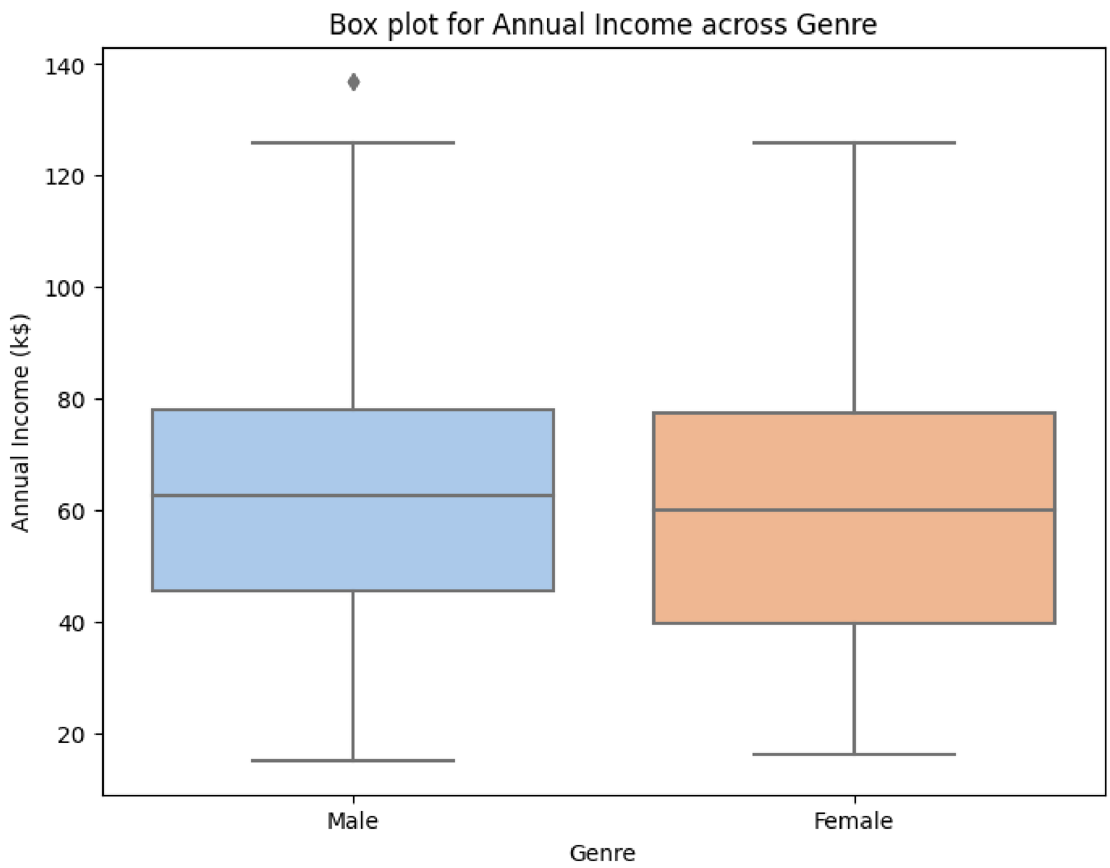
[13]:

```
df['Age_Group'] = pd.cut (df['Age'], bins =[0, 20, 30, 40, 50, 60, 100])
plt.figure(figsize=(8, 5))
sns.countplot(x ="Age_Group", data= df)
plt.title('Count of Customers in Age Groups')
plt.xlabel("Age Group")
plt.ylabel("Count")
plt.show()
```



[14]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot (x= "Annual Income (k$)", y='Spending Score (1-100)',␣
 ↪hue='Genre', data = df)
plt.title('Scatter plot for Annual Income and Spending Score')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```

[15]:
```
plt.figure(figsize=(8, 6))
sns.boxplot (x='Genre', y="Annual Income (k$)", data = df, palette='pastel')
plt.title('Box plot for Annual Income across Genre')
plt.xlabel('Genre')
plt.ylabel('Annual Income (k$)')
plt.show()
```



2.6 Model building

2.7 K- Mean clustering

[16]:
```
df.columns
```

[16]:
```
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)', 'Age_Group'],
      dtype='object')
```

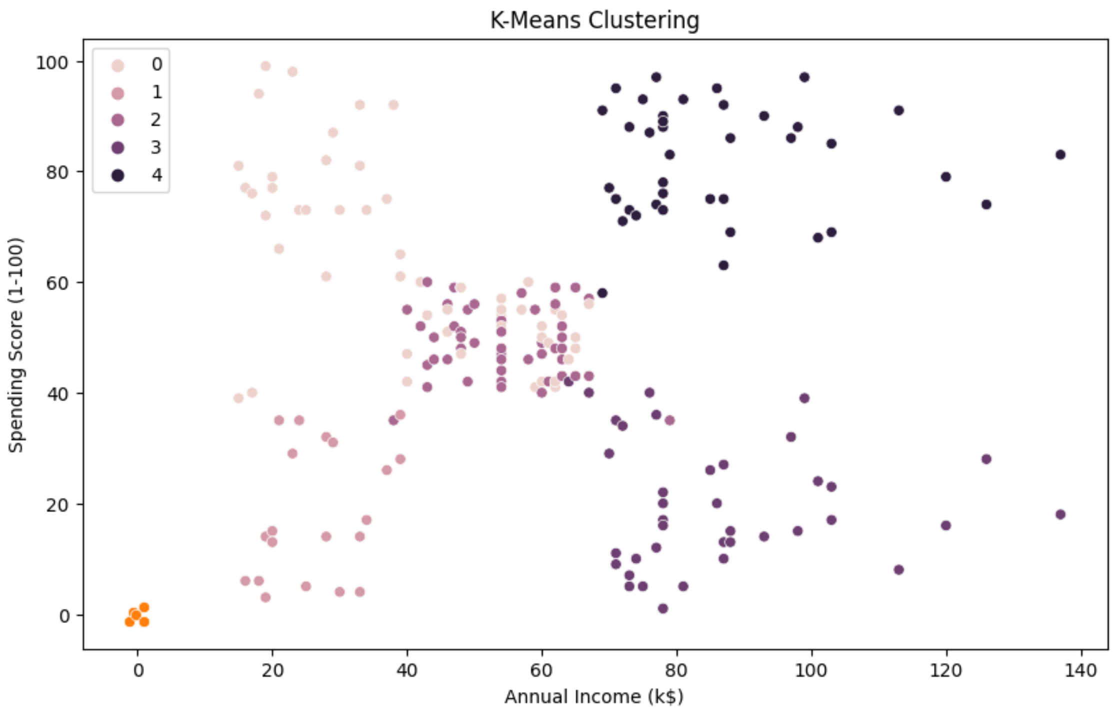
[17]:
```
X = df[['Age', 'Annual Income (k$)','Spending Score (1-100)']]
```

[18]:
```
scalar = StandardScaler()
X_scaled = scalar.fit_transform(X)
```

[19]:
```
kmeans = KMeans(n_clusters=5,random_state=77)
kmeans_label =kmeans.fit_predict(X_scaled)
```

[20]:
```
df['KMeans_Cluster'] = kmeans_label
```

[21]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot (x ='Annual Income (k$)', y=  "Spending Score (1-100)",␣
 ↪hue='KMeans_Cluster',data = df)
sns.scatterplot(x = kmeans.cluster_centers_[:, 1], y= kmeans.cluster_centers_[:
 ↪, 2])
plt.title("K-Means Clustering")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```



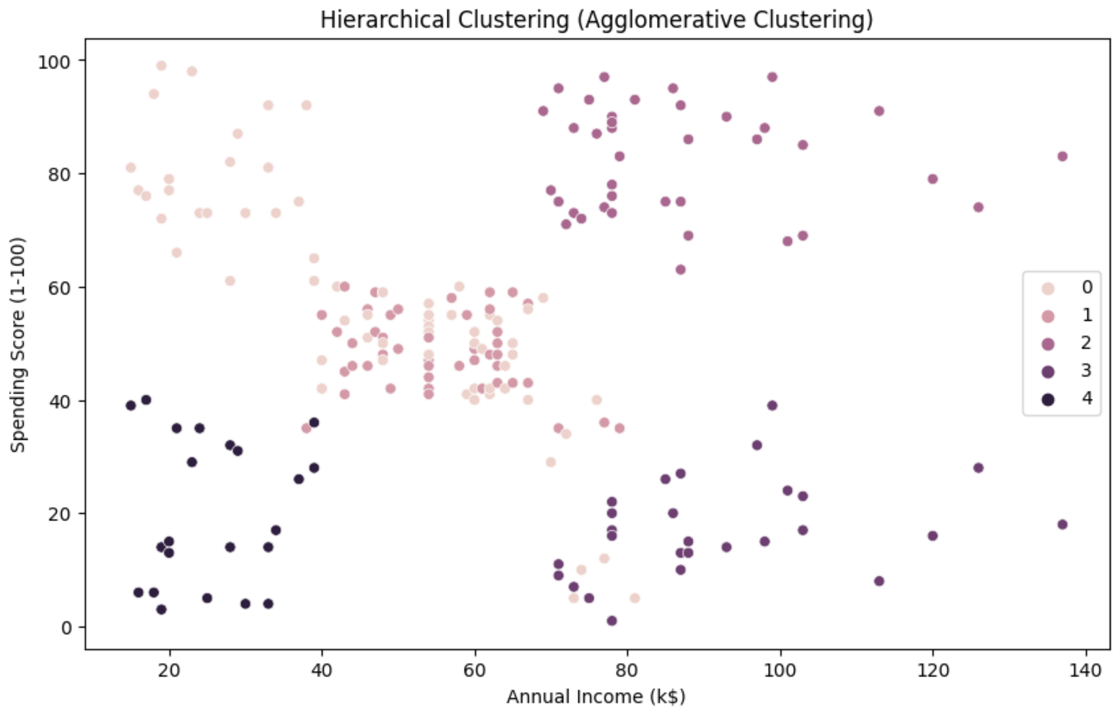
2.8 Hierarchical Clusturing

[23]:
```
agg_clustering =  AgglomerativeClustering (n_clusters=5)
agg_labels = agg_clustering.fit_predict(X_scaled)
```

[24]:
```
df['Agg_Cluster'] = agg_labels
```

[27]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',␣
 ↪hue='Agg_Cluster', data = df)
plt.title("Hierarchical Clustering (Agglomerative Clustering)")
plt.xlabel('Annual Income (k$)')
```

```
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```

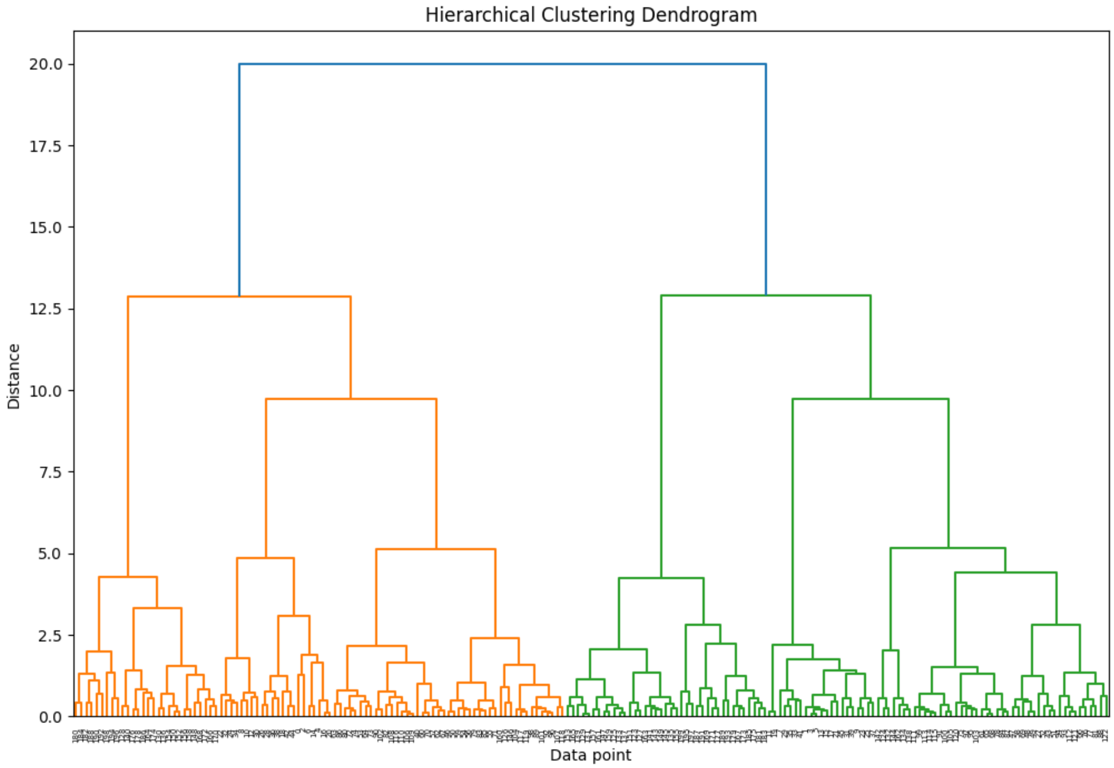


[30]:

```
Z =linkage (X_scaled, method="ward")
```

[32]:

```
plt.figure(figsize=(12, 8))
dendrogram(Z)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Data point')
plt.ylabel('Distance')
plt.show()
```

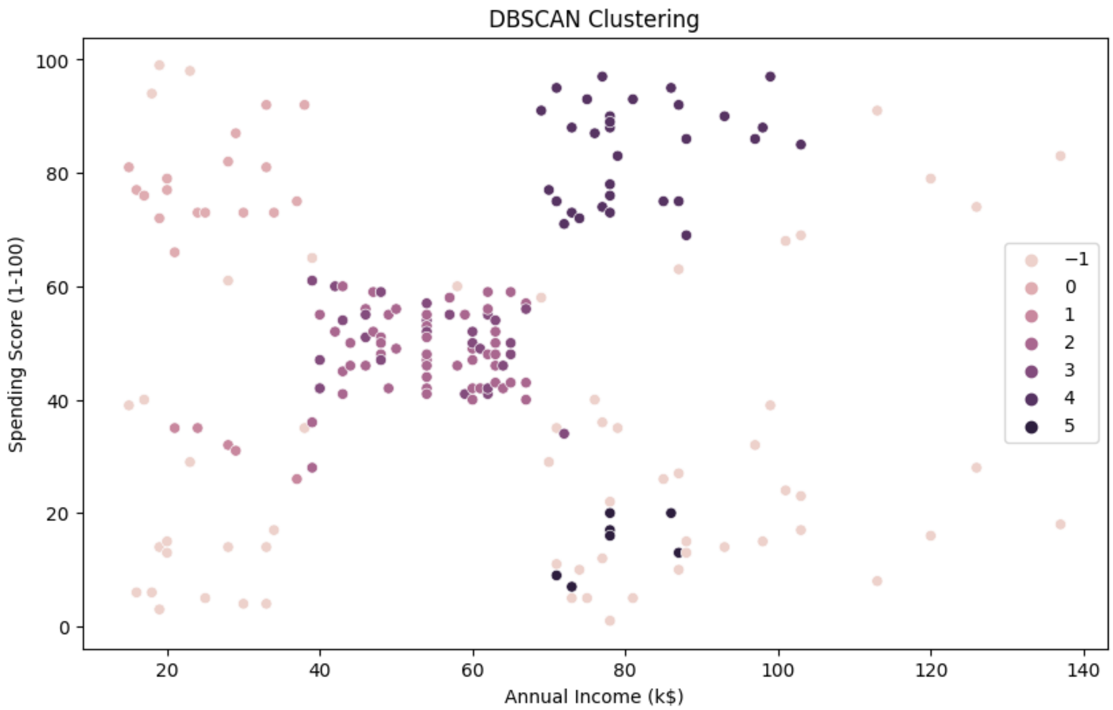


2.9 DBSCAN

[47]:
```
dbscan = DBSCAN (eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled)
```

[48]:
```
df['DBSCAN_Cluster'] = dbscan_labels
```

[50]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',␣
↪hue='DBSCAN_Cluster', data = df)
plt.title('DBSCAN Clustering')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

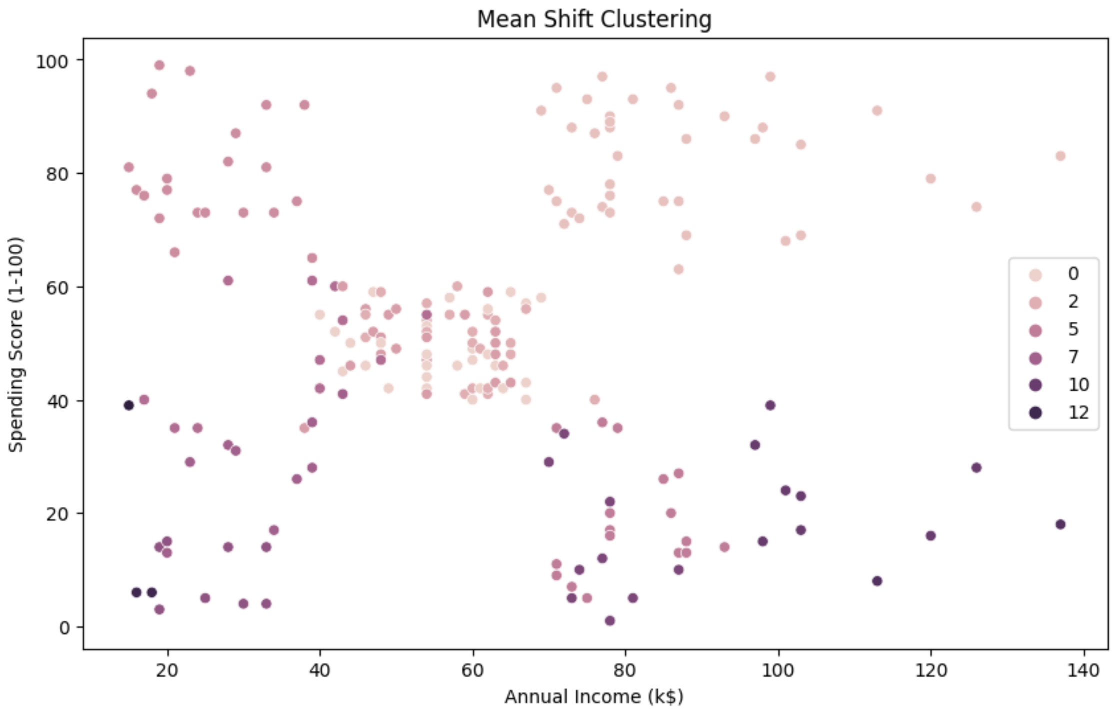


2.10 Mean Shift Clustering

[53]:
```
mean_shift = MeanShift(bandwidth=0.85)
mean_shift_labels = mean_shift.fit_predict(X_scaled)
```

[55]:
```
df['MeanShift_Cluster'] = mean_shift_labels
```

[58]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot (x='Annual Income (k$)', y='Spending Score (1-100)',␣
 ↪hue='MeanShift_Cluster',data = df)
plt.title('Mean Shift Clustering')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

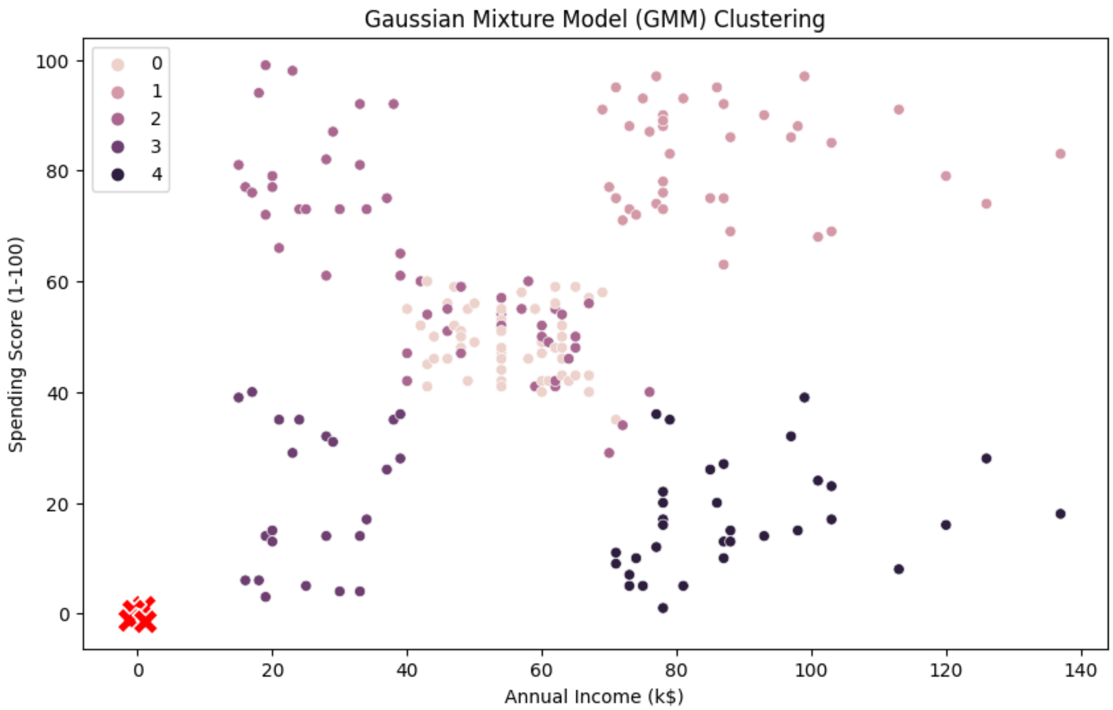


2.11 GMM Clusturing

[60]:
```
gmm = GaussianMixture(n_components=5, random_state=42)
gmm_labels= gmm.fit_predict(X_scaled)
```

[62]:
```
df['GMM_Cluster'] = gmm_labels
```

[68]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Annual Income (k$)', y= 'Spending Score (1-100)',␣
 ↪hue='GMM_Cluster',data = df)
sns.scatterplot(x=gmm.means_[:, 1], y=gmm.means_[:, 2], color='red', s=200,␣
 ↪marker='X')
plt.title('Gaussian Mixture Model (GMM) Clustering')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

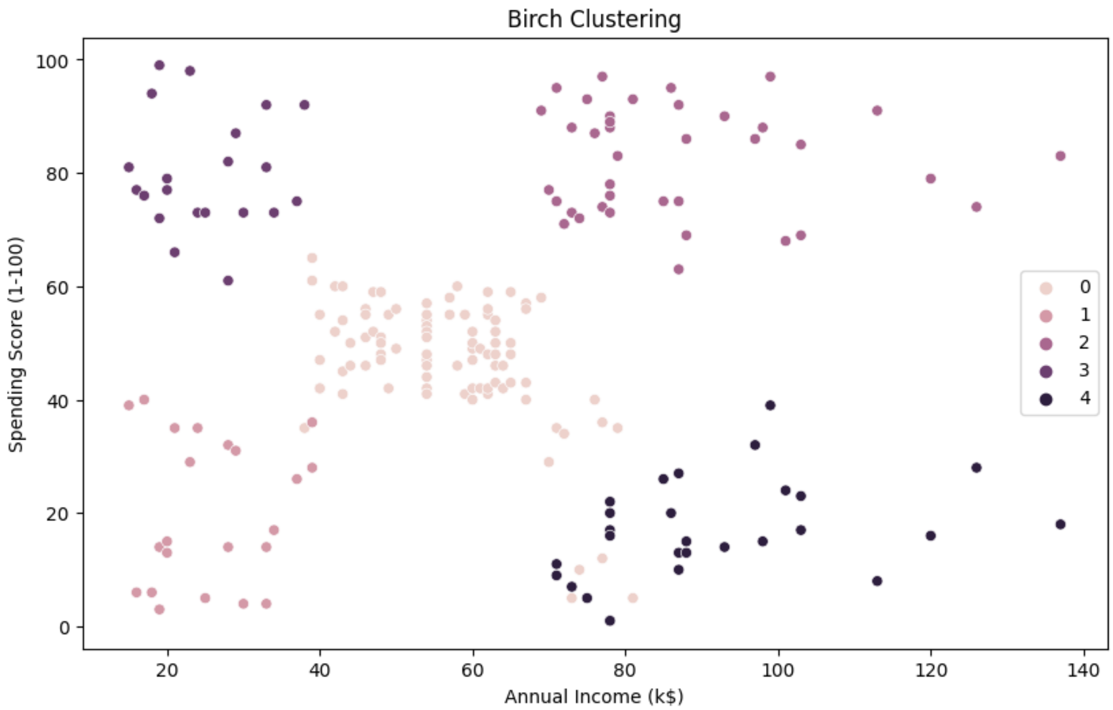


2.12 Birch

[71]:
```
birch = Birch(n_clusters=5)
birch_labels= birch.fit_predict(X_scaled)
```

[73]:
```
df['Birch_Cluster'] = birch_labels
```

[77]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',␣
  ↪hue='Birch_Cluster',data = df)
plt.title('Birch Clustering')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```


Birch Clustering
Spending Score (1-100)
Annual Income (k$)
0
1
2
3
4
0
20
40
60
80
100
20
40
60
80
100
120
140

```
[ ]:
```