week4task

April 5, 2025

0.1 Week 4: Customer Segmentation using K-Means Clustering

- Objective: Use unsupervised learning to segment customers into different groups based on purchasing behaviors.
- Skills: K-Means Clustering, Data Preprocessing, Cluster Analysis.

1 1. Data Preprocessing

Import Libraries:

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

Load and Inspect Data:

[3]:
```
df = pd.read_csv("Mall_Customers.csv")
print(df.head())
print(df.info())
```

```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
```

```
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```

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

[5]:

	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

Handle Missing Values:

[6]:
```
print(df.isnull().sum())  # Check for missing data
df.drop("CustomerID", axis=1, inplace=True)  # Drop irrelevant column
```

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

Encode Categorical Variables:

[7]:
```
df["Gender"] = df["Gender"].map({"Male": 0, "Female": 1})
```

Feature Scaling:

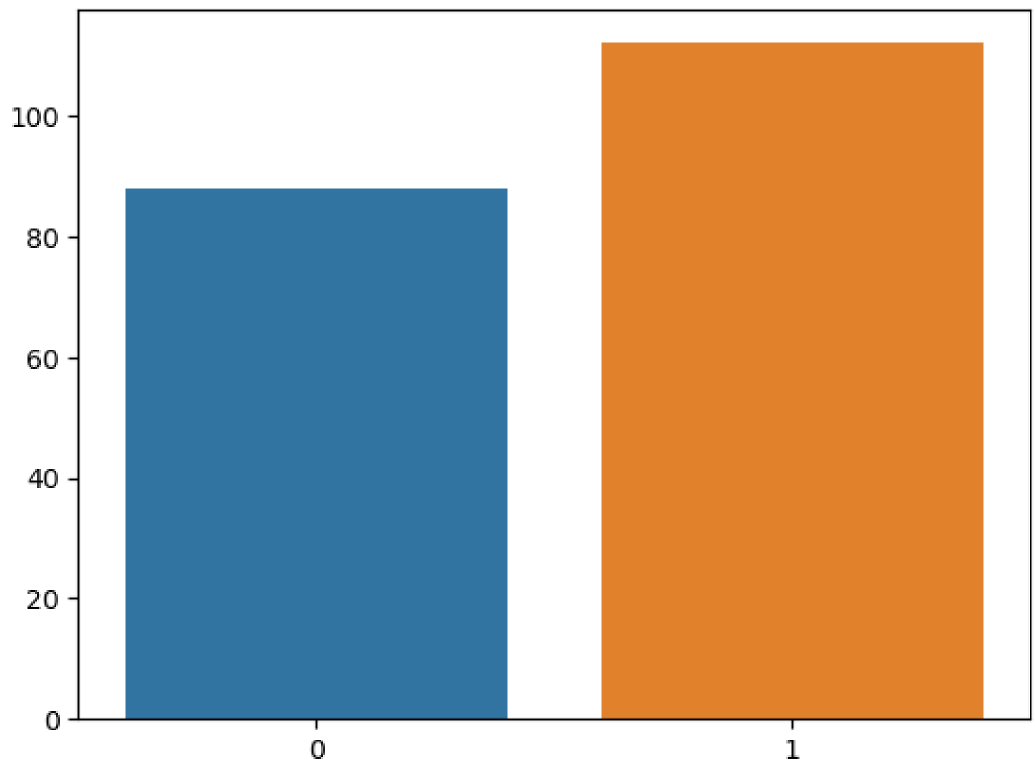
[8]:
```
scaler = StandardScaler()
X = df[["Annual Income (k$)", "Spending Score (1-100)"]]
X_scaled = scaler.fit_transform(X)
```

2 2. Exploratory Data Analysis (EDA)

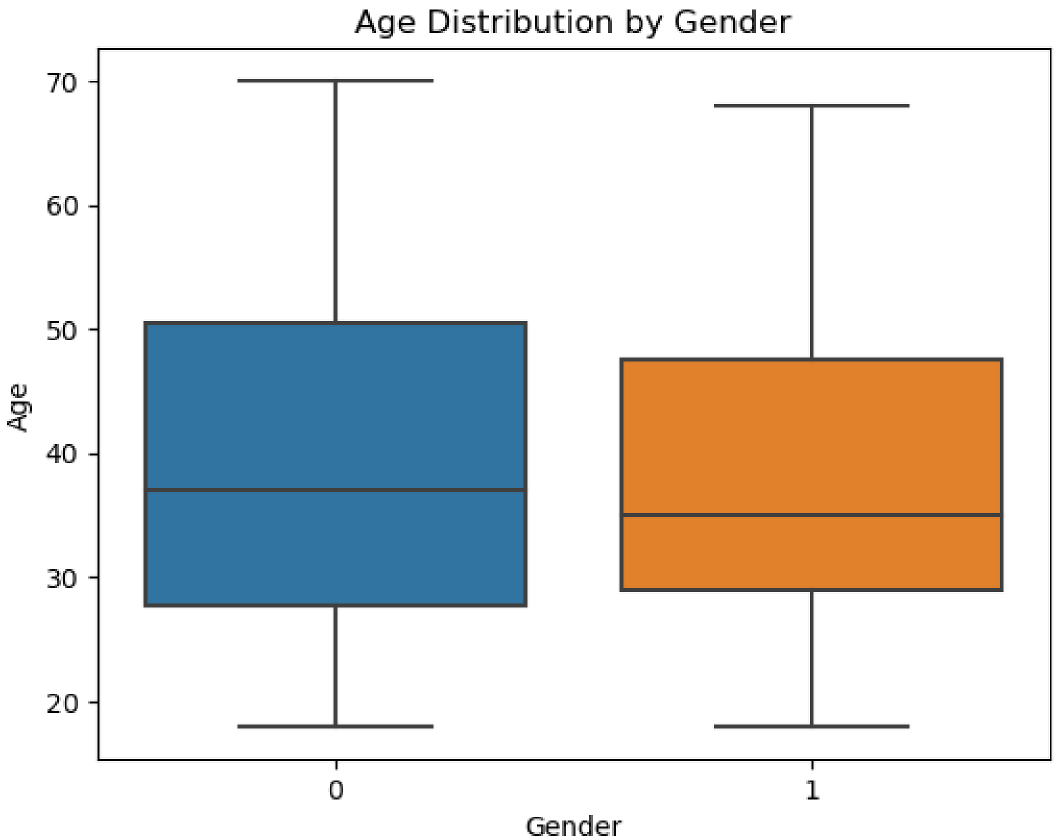
Distributions:

[14]:
```
sns.barplot(df['Gender'].value_counts().index,df['Gender'].value_counts().
 ↪values);
```

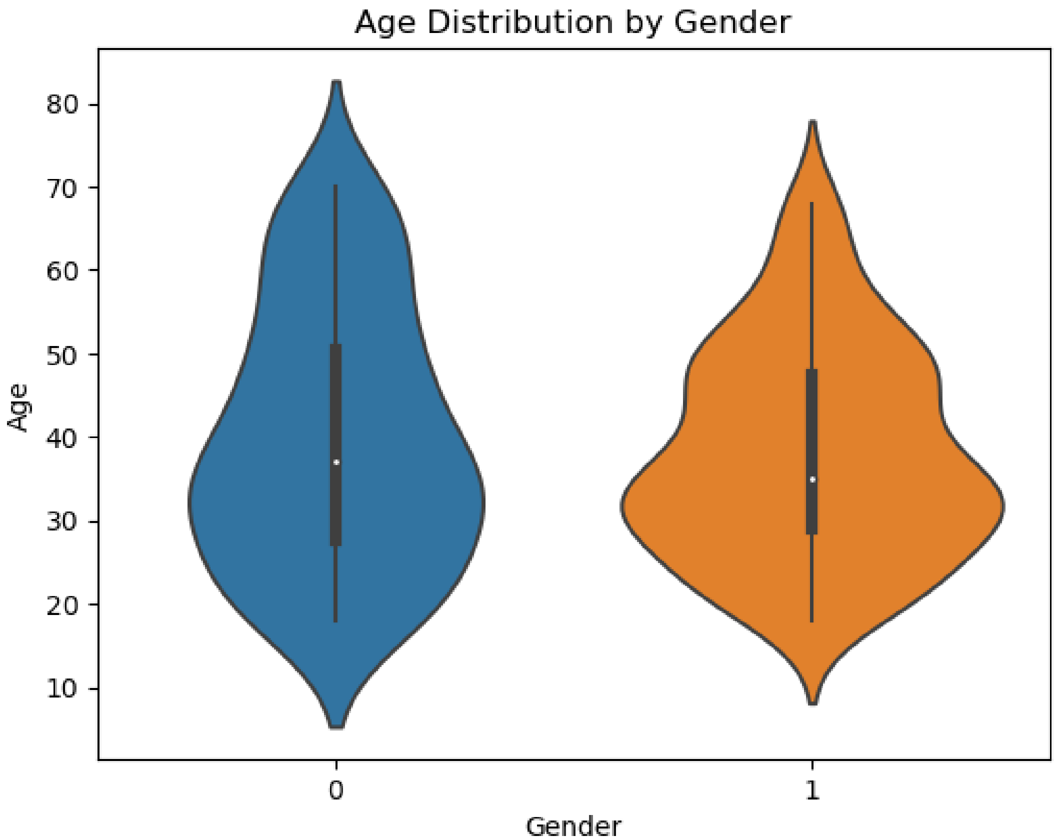
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



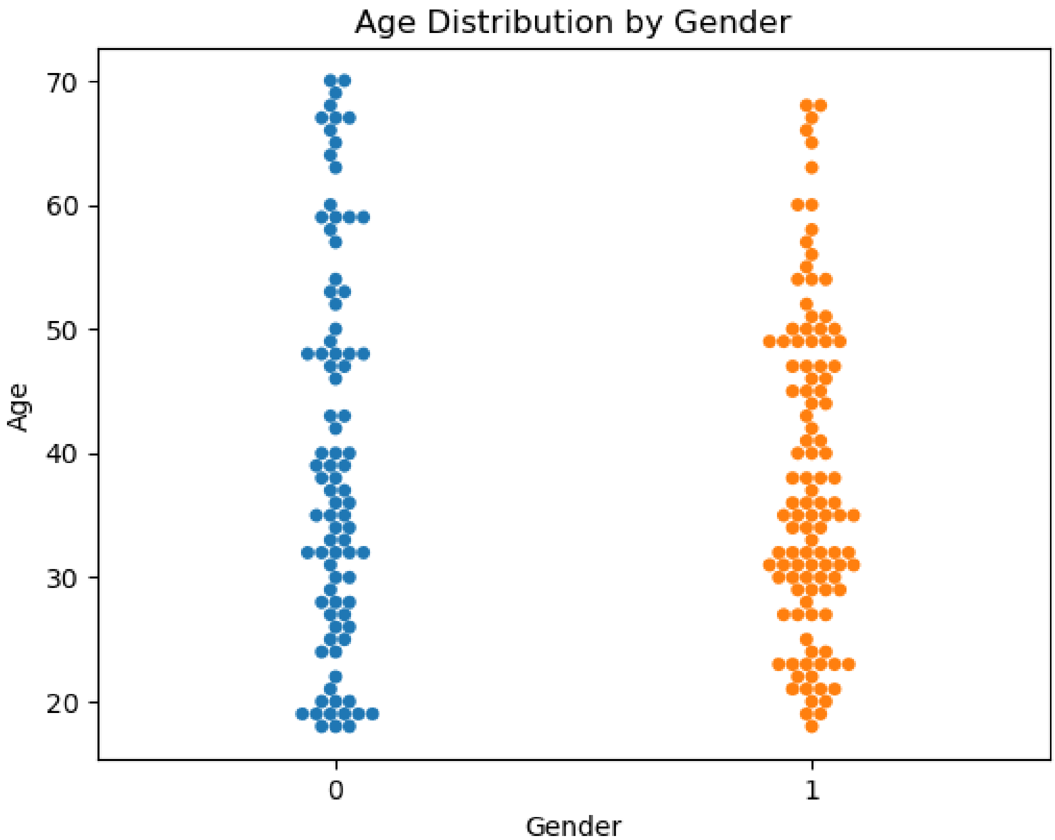
[15]:
```
sns.boxplot(x=df['Gender'], y=df['Age'])
plt.title("Age Distribution by Gender")
plt.show()
```



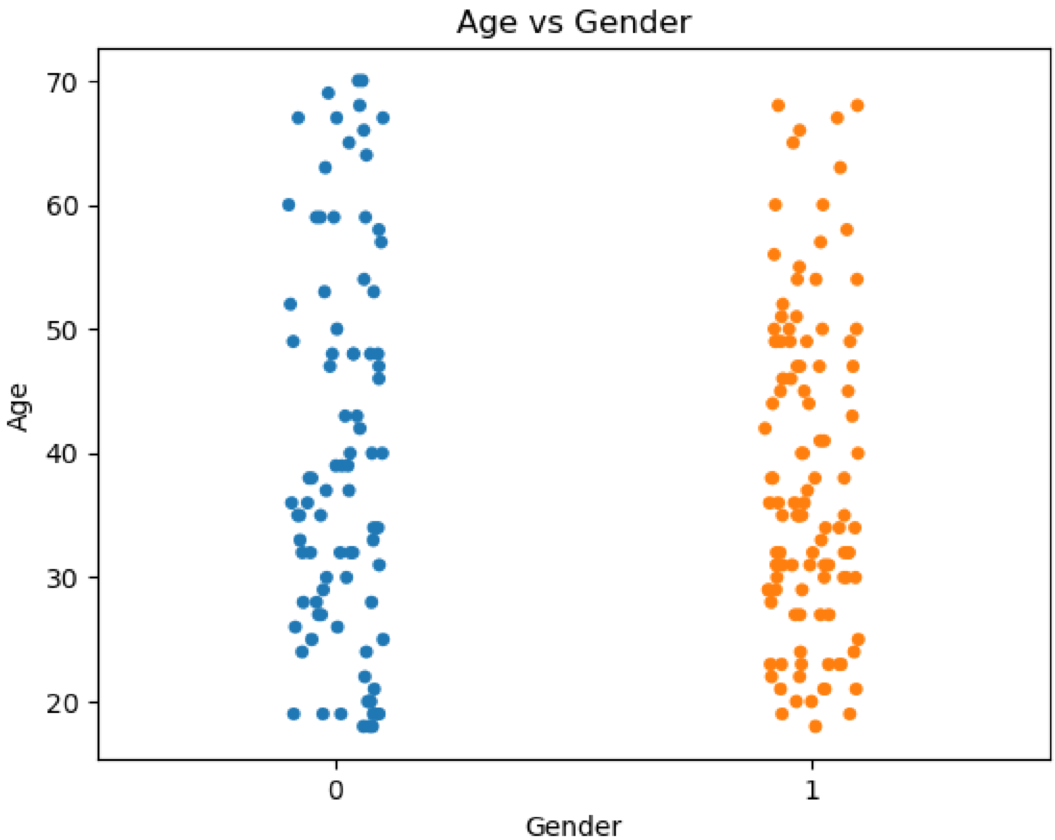
[16]:
```
sns.violinplot(x=df['Gender'], y=df['Age'])
plt.title("Age Distribution by Gender")
plt.show()
```



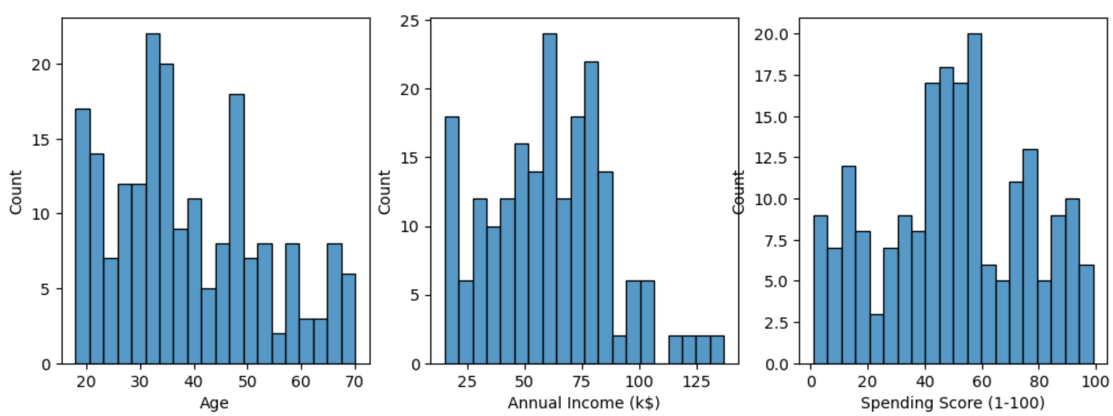
[17]:
```
sns.swarmplot(x=df['Gender'], y=df['Age'])
plt.title("Age Distribution by Gender")
plt.show()
```



[18]:
```
sns.stripplot(x=df['Gender'], y=df['Age'], jitter=True)
plt.title("Age vs Gender")
plt.show()
```



[38]:
```
plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
sns.histplot(df["Age"], bins=20)
plt.subplot(1, 3, 2)
sns.histplot(df["Annual Income (k$)"], bins=20)
plt.subplot(1, 3, 3)
sns.histplot(df["Spending Score (1-100)"], bins=20)
plt.show()
```



Correlation Analysis:

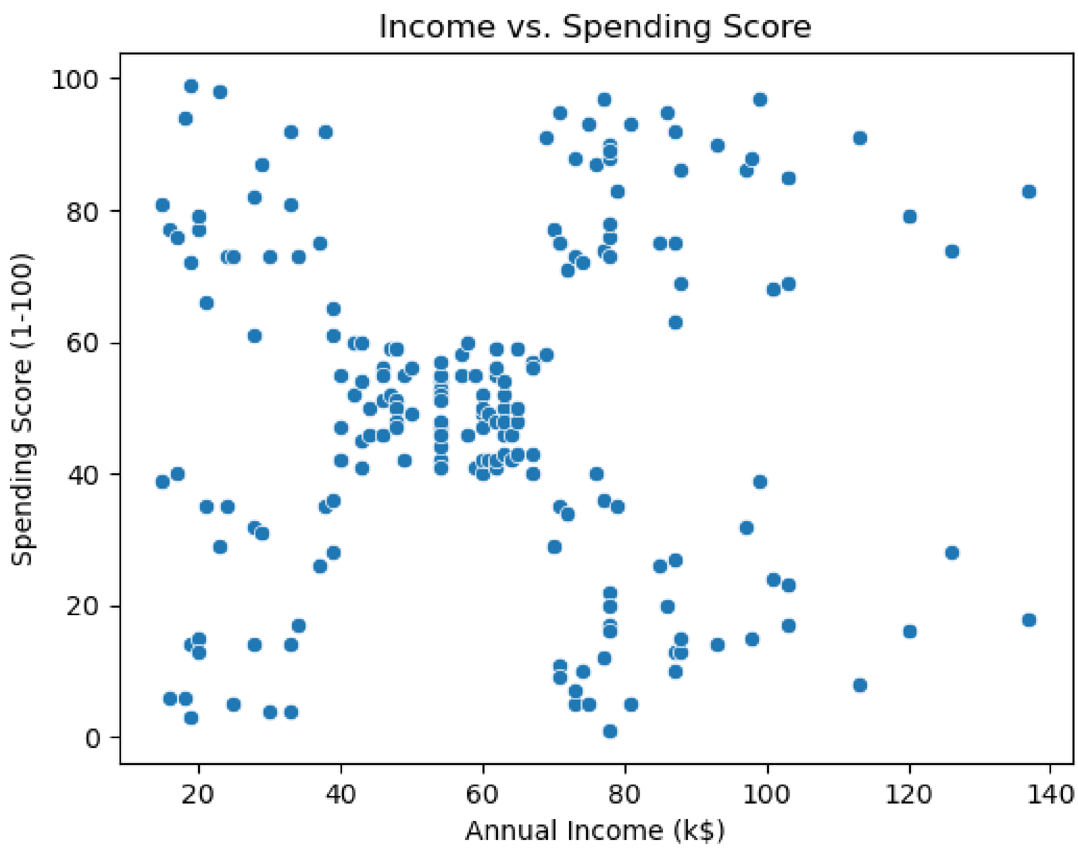
[39]:
```
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Matrix")
```

[39]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```

Income vs. Spending Score Scatter Plot:

[40]:
```
sns.scatterplot(data=df, x="Annual Income (k$)", y="Spending Score (1-100)")
plt.title("Income vs. Spending Score")
```

[40]:
```
Text(0.5, 1.0, 'Income vs. Spending Score')
```



3 3. K-Means Clustering

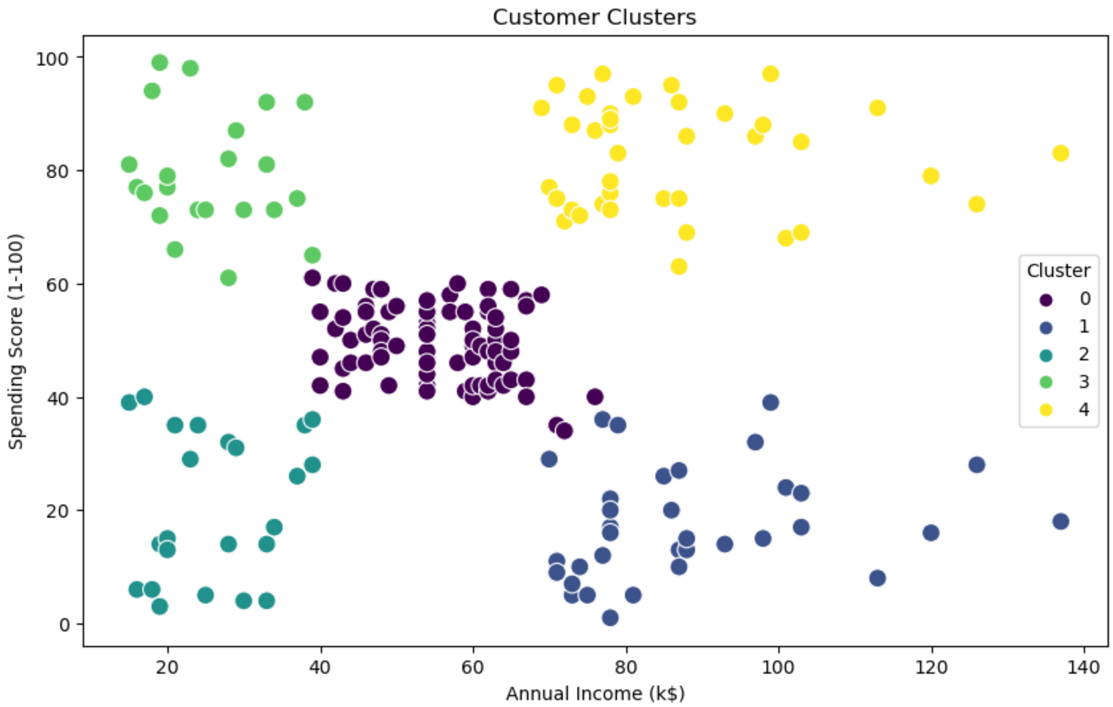
Apply K-Means with k=5:

[43]:
```
kmeans = KMeans(n_clusters=5, random_state=42)
df["Cluster"] = kmeans.fit_predict(X_scaled)
```

Visualize Clusters:

[44]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x="Annual Income (k$)", y="Spending Score (1-100)",
                hue="Cluster", palette="viridis", s=100)
plt.title("Customer Clusters")
```

[44]:
```
Text(0.5, 1.0, 'Customer Clusters')
```



4 4. Cluster Analysis

Cluster Characteristics:

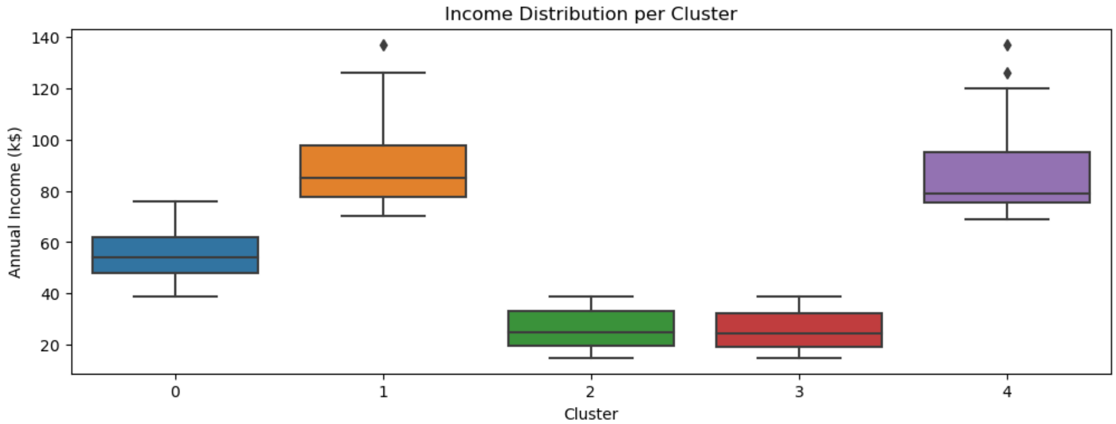
[45]:
```
cluster_summary = df.groupby("Cluster").agg({
    "Age": "mean",
    "Annual Income (k$)": "mean",
    "Spending Score (1-100)": "mean",
    "Gender": "mean"   # 1=Female, 0=Male
})
print(cluster_summary)
```

```
               Age  Annual Income (k$)  Spending Score (1-100)    Gender
Cluster
0        42.716049           55.296296               49.518519  0.592593
1        41.114286           88.200000               17.114286  0.457143
2        45.217391           26.304348               20.913043  0.608696
3        25.272727           25.727273               79.363636  0.590909
4        32.692308           86.538462               82.128205  0.538462
```

Boxplots for Age/Income/Spending Distributions:

[46]:
```
plt.figure(figsize=(12, 4))
sns.boxplot(data=df, x="Cluster", y="Annual Income (k$)")
plt.title("Income Distribution per Cluster")
```

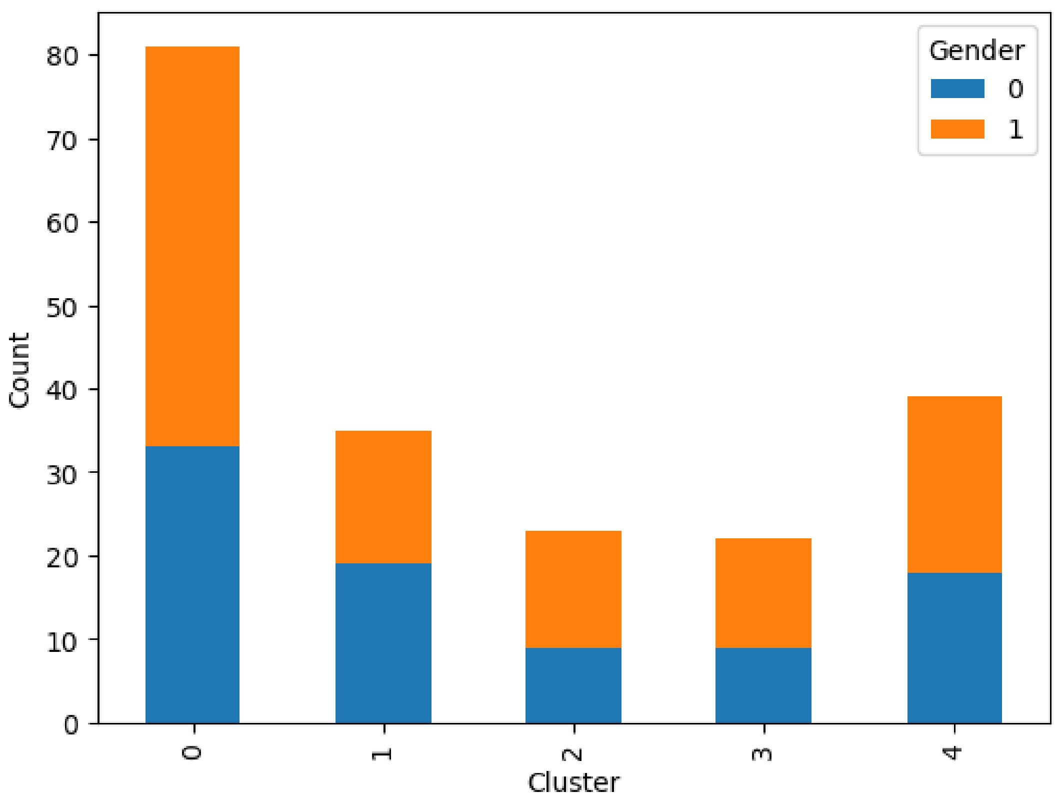
[46]: Text(0.5, 1.0, 'Income Distribution per Cluster')



Gender Distribution:

[47]:
```
pd.crosstab(df["Cluster"], df["Gender"]).plot(kind="bar", stacked=True)
plt.xlabel("Cluster")
plt.ylabel("Count")
```

[47]: Text(0, 0.5, 'Count')



```
[ ]:
```