Mall Customer Segmentation using K-means Clustering

Introduction

In this project, we will explore mall customer segmentation using K-means clustering, a powerful unsupervised machine learning technique. Mall customer segmentation involves dividing shoppers into distinct groups based on their characteristics and behaviors. This analysis helps mall managers and retailers better understand their customer base, tailor marketing strategies, and optimize store layouts and product offerings.

We'll be working with a dataset containing information about mall customers, including their age, gender, annual income, and spending score. By applying K-means clustering to this data, we aim to identify meaningful customer segments that can provide valuable insights for business decision-making.

Throughout this notebook, we'll go through the following steps:

1. Data loading and exploration
2. Data preprocessing and feature selection
3. Implementing K-means clustering
4. Visualizing the results
5. Interpreting the customer segments

By the end of this analysis, we'll have a better understanding of the different types of customers visiting the mall, which can inform targeted marketing campaigns and improve overall customer satisfaction.

---

1. Data Loading and Exploration

Let's start by loading the dataset and exploring its structure.

In [ ]:
```python
# import the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.cluster import KMeans
```

```
import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [ ]:
```
df = pd.read_csv('Mall_Customers.csv')
df.head()
```

Out[ ]:

	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

In [ ]:
```
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   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
```

Insights:

1. The dataset contains 200 rows (customers) and 5 columns.
2. All columns have non-null values, indicating no missing data.
3. The columns and their data types are:
   - CustomerID: int64
   - Age: int64
   - Annual Income (k$): int64
   - Spending Score (1-100): int64
4. The 'Gender' column is the only non-numeric column, suggesting it may need encoding for certain analyses.

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

Out[ ]:

	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

Insights:

1. CustomerID: Ranges from 1 to 200, confirming 200 unique customers.
2. Age:
   - Ranges from 18 to 70 years old.
   - Mean age is about 39 years.
3. Annual Income:
   - Ranges from $15,000 to$137,000.
   - Mean income is $60,560.
4. Spending Score:
   - Ranges from 1 to 99.
   - Mean score is 50.2.
5. The standard deviations suggest considerable variability in age, income, and spending scores.

In [ ]:

```
df.isnull().sum()
```

Out[ ]:

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

Insights:

1. There are no missing values in the dataset.

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

Out[ ]:
```
0
```

Insights:

1. There are no duplicate values in the dataset.

---

2. Data Preprocessing and Feature Selection

Before applying K-means clustering, we need to preprocess the data and select the relevant features.

In [ ]:
```
# check for outliers in the numerical columns

# Create subplots
fig = make_subplots(rows=2, cols=2, subplot_titles=('Age', 'Annual Income (k$)', 'S

# Add box plots
fig.add_trace(go.Box(y=df['Age'], name='Age'), row=1, col=1)
fig.add_trace(go.Box(y=df['Annual Income (k$)'], name='Annual Income'), row=1, col=
fig.add_trace(go.Box(y=df['Spending Score (1-100)'], name='Spending Score'), row=2,

# Update layout
fig.update_layout(height=800, width=1000, showlegend=False)

# Show the plot
fig.show()
```

Insights:

1. **Age:** The boxplot shows a relatively symmetric distribution with a median around 35-40 years. There are a few mild outliers on the upper end, representing older customers.
2. **Annual Income:** The income distribution is slightly right-skewed with a median around 70k$. There are several high-income outliers, indicating some customers with significantly higher incomes than the majority.
3. **Spending Score:** This feature shows a fairly symmetric distribution with a median around 50. There are no significant outliers, suggesting that spending behaviors are relatively consistent across the customer base.

4. Overall, while there are some mild outliers in age and more pronounced ones in income, they don't appear to be extreme enough to warrant removal. These outliers might represent important customer segments.

In [ ]:
```
# checkking the distribution of the numerical columns

# Create subplots
fig = make_subplots(rows=2, cols=2, subplot_titles=('Age Distribution', 'Annual Inc

# Add histogram traces
fig.add_trace(go.Histogram(x=df['Age'], name='Age'), row=1, col=1)
fig.add_trace(go.Histogram(x=df['Annual Income (k$)'], name='Annual Income'), row=1
fig.add_trace(go.Histogram(x=df['Spending Score (1-100)'], name='Spending Score'),

# Update layout
fig.update_layout(height=800, width=1200, showlegend=False)

# Show the plot
fig.show()
```

Insights:

1. **Age Distribution:** The age column appears to be approximately normally distributed, with a peak around the middle ages and tapering off towards younger and older ages. This suggests a balanced representation of different age groups in the customer base.

2. **Annual Income Distribution:** The annual income column shows a right-skewed distribution. This indicates that while most customers have incomes clustered around a lower to middle range, there are some high-income outliers pulling the distribution to the right. This is common in income data and suggests a diverse customer base in terms of purchasing power.

3. **Spending Score Distribution:** The spending score column appears to be relatively uniformly distributed across the range. This suggests that customers in this dataset have a wide variety of spending behaviors, from low to high, without any particular spending pattern being dominant.

In [ ]:
```
# encoding the gender column
df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
df.head()
```

Out[ ]:

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

In [ ]:

```
# checking the correlation matrix

corr_matrix = df.corr()

fig = go.Figure(data=go.Heatmap(
    z=corr_matrix.values,
    x=corr_matrix.columns,
    y=corr_matrix.index,
    colorscale='RdBu',
    zmin=-1,
    zmax=1,
    text=corr_matrix.values.round(2),
    texttemplate='%{text}',
    textfont={"size": 10},
    hoverongaps=False
))

fig.update_layout(
    title='Correlation Matrix',
    width=800,
    height=600
)

fig.show()
```

Insights:

1. There is a weak positive correlation (0.19) between Age and Annual Income, suggesting that older customers tend to have slightly higher incomes.
2. There is a weak negative correlation (-0.33) between Age and Spending Score, indicating that younger customers tend to have slightly higher spending scores.
3. There is no significant correlation between Annual Income and Spending Score (0.01), suggesting that income doesn't necessarily predict spending behavior.
4. Gender shows very weak correlations with other variables, implying that gender may not be a strong factor in determining income or spending patterns.

3. Implementing K-means Clustering

Now, let's implement K-means clustering to identify customer segments based on their annual income and spending score.

In [ ]:
```
# Select the features for clustering
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Standardize the features
X_scaled = (X - X.mean()) / X.std()
```

In [ ]:
```
# plotting elbow method to find the optimal number of clusters

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

In [ ]:
```
# implementing k-means clustering
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X_scaled)
y_pred = kmeans.predict(X_scaled)
```

In [ ]:
```
# Add the cluster labels to the original dataframe
df['Cluster'] = y_pred
```

---

4. Visualizing the results

In [ ]:
```
fig = go.Figure(data=go.Scatter(x=list(range(1, 11)), y=wcss, mode='lines+markers'))

fig.update_layout(
    title='Elbow Method For Optimal k',
    xaxis_title='Number of clusters',
    yaxis_title='WCSS',
    width=1000,
    height=600
)

fig.show()
```

`Insights:`

1. The elbow curve shows a sharp decrease in WCSS up to 5 clusters.

2. After 5 clusters, the decrease in WCSS becomes more gradual.
3. This suggests that the optimal number of clusters is likely 5.
4. Using 5 clusters balances between minimizing within-cluster variance and avoiding overfitting.
5. However, it's worth noting that the "elbow" is not extremely pronounced, so exploring 4 or 6 clusters might also be valuable.

In [ ]:
```
# plotting the clusters

fig = px.scatter(df, x='Annual Income (k$)', y='Spending Score (1-100)', color='Clu
fig.update_layout(
    title='K-means Clustering',
    xaxis_title='Annual Income (k$)',
    yaxis_title='Spending Score (1-100)',
    width=1000,
    height=600
)
fig.show()
```

---

5. Interpreting the customer segments

In [ ]:
```
# Interpret the customer segments
cluster_means = df.groupby('Cluster').mean()
cluster_means
```

Out[ ]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
Cluster					
0	86.320988	0.592593	42.716049	55.296296	49.518519
1	162.000000	0.538462	32.692308	86.538462	82.128205
2	23.090909	0.590909	25.272727	25.727273	79.363636
3	164.371429	0.457143	41.114286	88.200000	17.114286
4	23.000000	0.608696	45.217391	26.304348	20.913043

`Interpreting the Customer Segments`:

Based on the cluster means, we can interpret the customer segments as follows:

1. **Cluster 0: "Average Consumers"**

- Moderate income and moderate spending score
- These customers represent the average mall shopper

2. **Cluster 1: "High-Income, Low-Spenders"**

   - High annual income but low spending score
   - Potentially frugal or selective in their purchases

3. **Cluster 2: "Low-Income, High-Spenders"**

   - Low annual income but high spending score
   - May be impulsive buyers or highly engaged with mall offerings

4. **Cluster 3: "Target Customers"**

   - High annual income and high spending score
   - Prime customers for luxury goods and premium services

5. **Cluster 4: "Low-Income, Low-Spenders"**

   - Low annual income and low spending score
   - May be budget-conscious or infrequent mall visitors

These interpretations provide valuable insights for targeted marketing strategies and personalized customer experiences.