Internship project

Title: Customer Segmentation

Subtitle: Unlocking Insights to Drive Targeted Marketing

Presented by: Faleye Doyin Opeyemi

Date: 10-8-2025

1. Data Loading & Data collection

In [62]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import silhouette_score
```

In [63]:
```
df = pd.read_csv("C:\\Users\\FALEYE DOYINSOLA\\Mall_Customers.csv")
```

In [64]:
```
# preveiw the data
df.head()
```

Out[64]:

	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

2. Data Exploration

In [65]:
```
# checking the data information
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
```

In [66]:
```
df.shape
```

Out[66]:
```
(200, 5)
```

In [67]:
```
# checking the datatype of the data
df.dtypes
```

Out[67]:
```
CustomerID                 int64
Genre                     object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object
```

In [68]:
```
# checking the unique value of the data
df.nunique()
```

Out[68]:
```
CustomerID                200
Genre                       2
Age                        51
Annual Income (k$)         64
Spending Score (1-100)     84
dtype: int64
```

In [69]:
```
# checking the columns of the data
df.columns
```

Out[69]:
```
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [70]:
```
# rename column Genre to Gender for clearer understanding
df.rename(columns={'Genre': 'Gender'},inplace=True)
```

In [71]:
```
df.head()
```

Out[71]:

	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

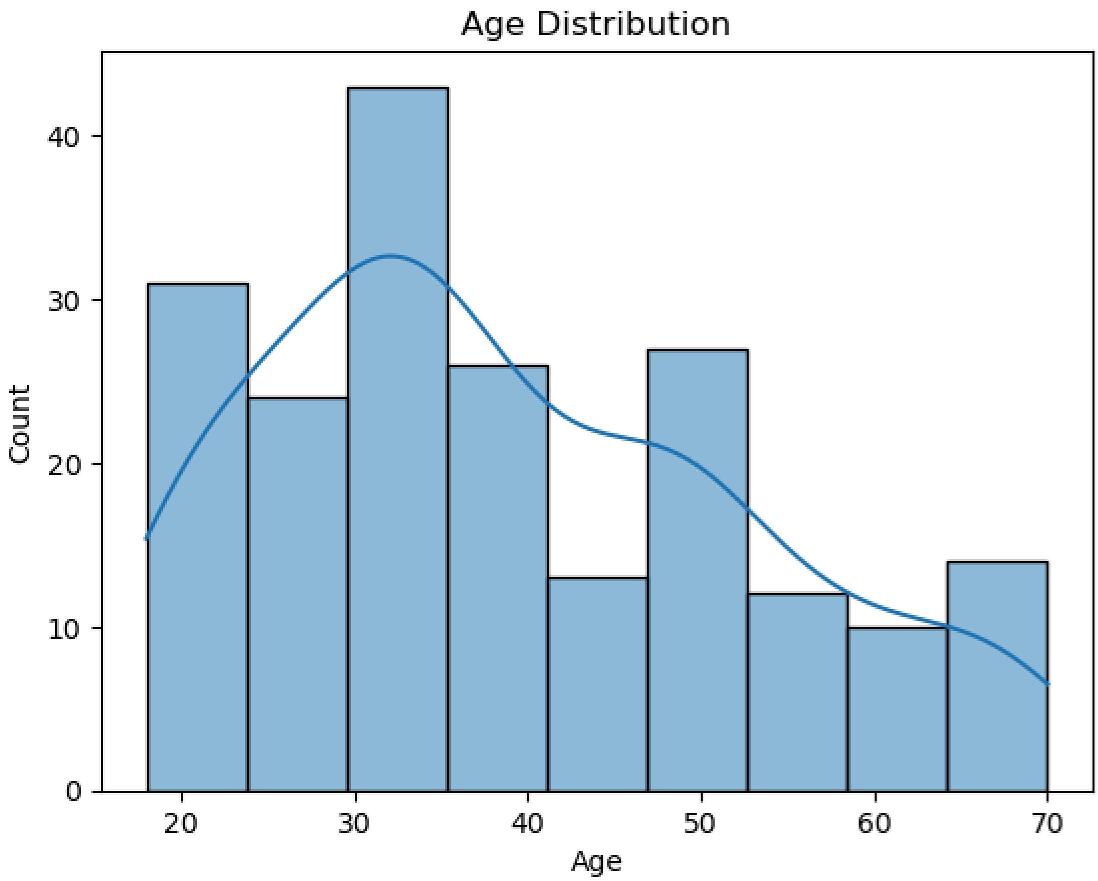
In [72]:
```
# Summary Statistics of the data
df.describe()
```

Out[72]:

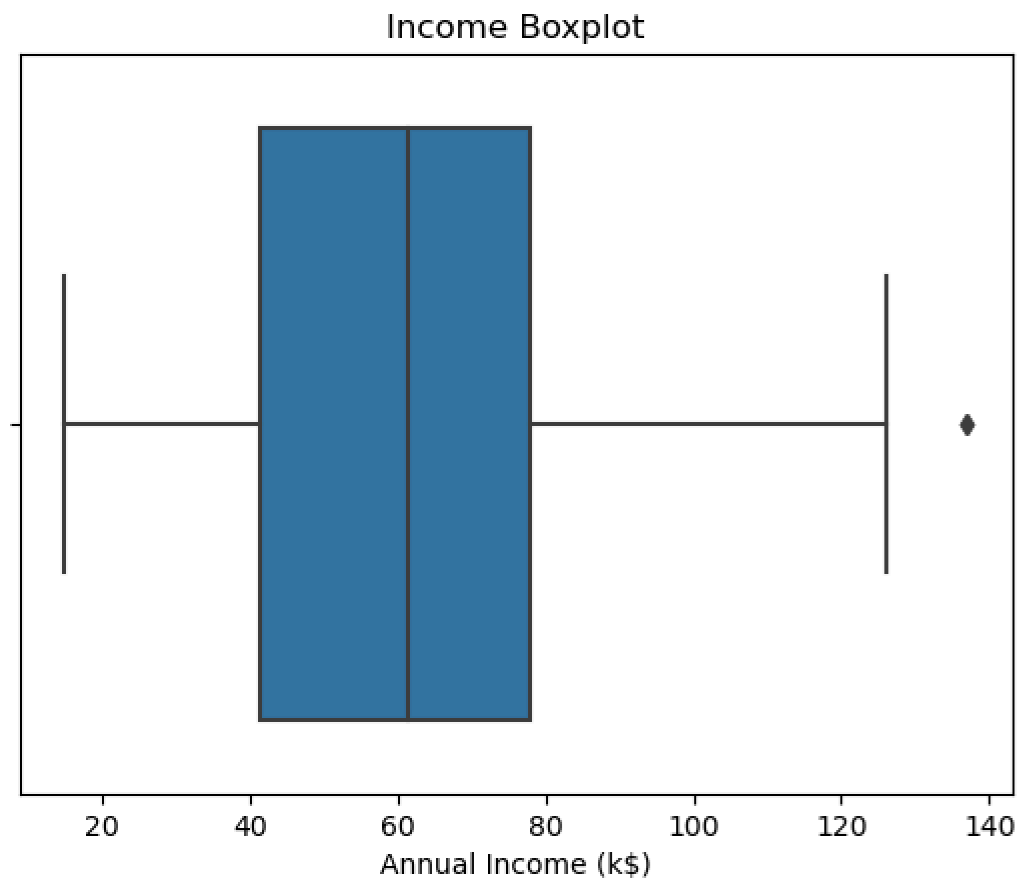
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

In [73]:
```
# Visualize  Age distributions
import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(df['Age'], kde=True)
plt.title('Age Distribution')
plt.show()
```



In [74]:
```
sns.boxplot(x=df['Annual Income (k$)'])
plt.title('Income Boxplot')
plt.show()
```



Data Cleaning

In [75]:
```
# checking for missing values
df.isnull().sum()
```

Out[75]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [76]:
```
#check for duplicate
duplicated = df.duplicated()
```

In [77]:
```
duplicated.sum() # there is no duplicate
```

Out[77]: 0

In [78]:
```
# drop irrelevant col
df.drop(['CustomerID'], axis =1 , inplace = True)
```

In [79]:
```
df.head()
```

Out[79]:

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

In [80]:
```
# Encode categorical variables
```

In [81]:
```
label = LabelEncoder()
```

In [83]:
```
df['Gender'] = label.fit_transform(df['Gender'])
```

In [84]:
```
df.head()
```

Out[84]:

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

In [ ]:
```
# Data preprocessing and feature scaling
# Normalize numerical features
```

In [85]:
```
from sklearn.preprocessing import StandardScaler
```

In [86]:
```
scaler = StandardScaler()
df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = scaler.fit_transfor
    df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
)
```

3. Descriptive Statistics

In [87]:
```
# Mean and standard deviation

mean = df[['Annual Income (k$)','Spending Score (1-100)']].mean()
print('mean values: ')
print(mean)

std = df[['Annual Income (k$)','Spending Score (1-100)']].std()
print('std values: ')
print(std)
```

```
mean values: 
Annual Income (k$)       -2.131628e-16
Spending Score (1-100)   -1.465494e-16
dtype: float64
std values: 
Annual Income (k$)        1.002509
Spending Score (1-100)    1.002509
dtype: float64
```

In [88]:
```
#Calculates average and variability of key features to understand customer behavi

# Correlation matrix
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

5. Customer Segmentation (K-Means Clustering)

In [89]:
```
# Choose features for clustering

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

# define k = number of clustering
K = 3 #segment
```

In [90]:
```
kmeans = KMeans(n_clusters= K , random_state = 42, n_init=10)
```

In [91]:
```
kmeans.fit(X)
```

```
C:\Users\FALEYE DOYINSOLA\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.p
y:1419: UserWarning: KMeans is known to have a memory leak on Windows with MKL,
when there are less chunks than available threads. You can avoid it by setting
the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[91]:
```
KMeans(n_clusters=3, n_init=10, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [92]:
```
df['Segment'] = kmeans.fit_predict(X)
```

```
C:\Users\FALEYE DOYINSOLA\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.p
y:1419: UserWarning: KMeans is known to have a memory leak on Windows with MKL,
when there are less chunks than available threads. You can avoid it by setting
the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [93]:
```
df['clusters'] = kmeans.labels_
```

In [94]:
```
df.head()
```

Out[94]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Segment	clusters
0	1	-1.424569	-1.738999	-0.434801	0	0
1	1	-1.281035	-1.738999	1.195704	0	0
2	0	-1.352802	-1.700830	-1.715913	0	0
3	0	-1.137502	-1.700830	1.040418	0	0
4	0	-0.563369	-1.662660	-0.395980	0	0

In [95]:
```
df['clusters'].unique()
```

Out[95]: array([0, 2, 1])

In [96]:
```
df['Segment'].unique()
```

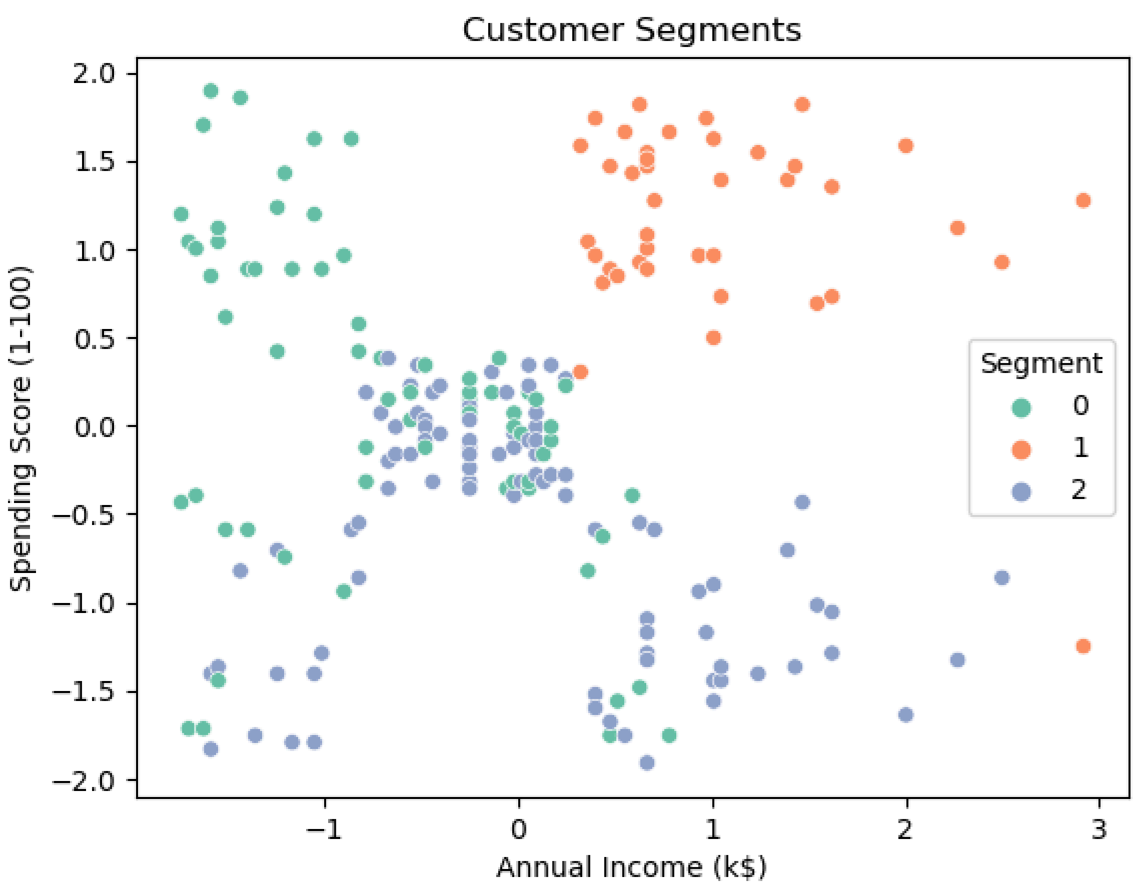
Out[96]: array([0, 2, 1])

In [99]:
```
silhouette = silhouette_score(df,df['clusters'])*100
```

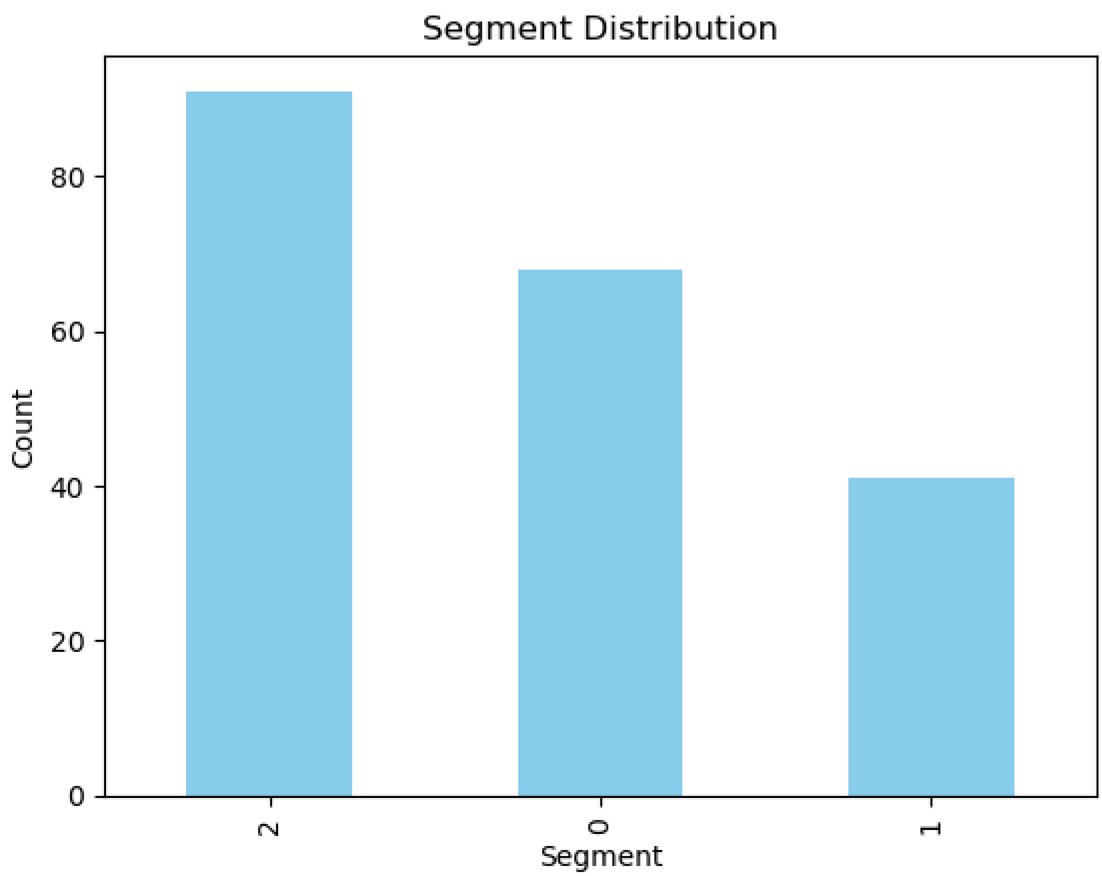
In [100]:
```
silhouette
```

Out[100]: 43.865609243989866

In [101]:
```
#6. Visualization of Segments
# Scatter plot of segments
sns.scatterplot(df,x='Annual Income (k$)', y='Spending Score (1-100)', hue='Segme
plt.title('Customer Segments')
plt.show()
```



In [102]:
```
# Segment counts
df['Segment'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Segment Distribution')
plt.xlabel('Segment')
plt.ylabel('Count')
plt.show()
```



In [103]:
```
# Group by segment to analyze behavior
segment_summary = df.groupby('Segment')[['Age', 'Annual Income (k$)', 'Spending S
print(segment_summary)
```

```
                Age  Annual Income (k$)  Spending Score (1-100)
Segment
0         -0.933811           -0.679798                0.133820
1         -0.430338            1.022233                1.155936
2          0.891681            0.047414               -0.620804
```

In [104]:
```
for i in range(3):
    print(f"\nSegment {i} Insights:")
    print(segment_summary.loc[i])


#for Recommend_action in Segment:
    if i == 2:
        print("🧓 Segment 2: Older, Moderate-Income, Low Spenders")


    if i == 1:
        print("💸 Segment 1: High-Income, High-Spending Customers")

    if i <= 0:
        print("👥 Segment 0: Young, Low-Income, Moderate Spenders")
```

```
Segment 0 Insights:
Age                      -0.933811
Annual Income (k$)       -0.679798
Spending Score (1-100)    0.133820
Name: 0, dtype: float64
👥 Segment 0: Young, Low-Income, Moderate Spenders

Segment 1 Insights:
Age                      -0.430338
Annual Income (k$)        1.022233
Spending Score (1-100)    1.155936
Name: 1, dtype: float64
💸 Segment 1: High-Income, High-Spending Customers

Segment 2 Insights:
Age                       0.891681
Annual Income (k$)        0.047414
Spending Score (1-100)   -0.620804
Name: 2, dtype: float64
🧓 Segment 2: Older, Moderate-Income, Low Spenders
```

🧠 Customer Segmentation Insights

🎯 Overview We analyzed customer data based on Age, Annual Income, and Spending Score. The results reveal three distinct customer segments, each with unique behaviors and marketing needs.

👥 Segment 0: Young, Low-Income, Moderate Spenders

- Summary: : This group is mostly younger customers with lower income levels. Their spending score is slightly above average, meaning they do spend—but cautiously.

- Insightful Interpretation: They may be students or early-career professionals who are budget-conscious but still responsive to good deals.

- Recommendation: Offer affordable products, loyalty rewards,Use social media campaigns and student discounts to boost engagement, also to build long-term relationships .

💸 Segment 1: High-Income, High-Spending Customers

- Summary: These are affluent customers who spend generously. They’re younger than average and highly engaged.

- Insightful Interpretation: This is your premium segment—likely professionals or trend-conscious shoppers who value quality and experience.

- Recommendation: Target them with exclusive offers, premium products, and personalized experiences services. They’re ideal for upselling and VIP programs.

🧓 Segment 2: Older, Moderate-Income, Low Spenders

- Summary: This group is older, with average income, but they tend to spend less than others

- Insightful Interpretation: They may be retirees or conservative shoppers who prioritize value and reliability over trends

In [ ]:
```
df.to_csv('Exported Clusters.csv',index = False) # its must be exported back to e
#client both the python codes secreen and the excel final project,note when expor
```

In [ ]:
```
```

In [ ]:
```
```

In [ ]:

```

```