CUSTOMER SEGMENTATION USING DATA SCIENCE

TEAM MEMBERS

T.TAMIL SELVAN

V.MUTHUMARIAPPAN

K.RAHUL

R.PANDIYARAJAN

A.ARUNKUMAR

PHASE 4 SUBMISSION DOCUMENT

PHASE 4: DEVELOPMENT PART 2



INTRODUCTION

* Today many of the businesses are going online and, in this case, online marketing is becoming essential to hold customers, but during this, considering all customers as same and targeting all of them with similar marketing strategy is not very efficient way rather it's also annoys the customers by neglecting his or her individuality, so customer segmentation is becoming very popular and also became the efficient solution for this existing problem. Customer segmentation is defined as dividing company's customers on the basis of demographic (age, gender, marital status) and behavioral (types of products ordered, annual income) aspects. Since demographic characteristics does not emphasize on individuality of customer because same age groups may have different interests so behavioral aspects is a better approach for customer segmentation as its focus on individuality and we can do proper segmentation with the help of it.

**FEATURE ENGINEERING**

Customer segmentation is a common application of data science in marketing and business. It involves grouping customers into distinct segments based on their behavior, characteristics, and preferences. The goal is to understand customer needs better and tailor marketing strategies to specific segments. Below, I'll provide you with a high-level overview of the process and share some program codes in Python for customer segmentation

**Steps for Customer Segmentation:**

**Data Collection:**

Gather relevant data on your customers. This may include demographic information, purchase history, website activity, and more.

**Data Preprocessing:**

Clean and preprocess the data. This includes handling missing values, encoding categorical variables, and scaling/normalizing numerical features.

**Feature Engineering:**

Create new features or transform existing ones to capture important information about customers.

**Segmentation:**

Use a clustering algorithm to group customers into segments based on their features and behavior.

**Evaluation:**

Assess the quality of your segments using appropriate metrics (e.g., silhouette score, Davies-Bouldin index).

**Interpretation:**

Interpret the characteristics and behavior of each segment to gain insights into your customer base**.**

**Marketing Strategy:**

Develop targeted marketing strategies for each segment.

Python Code for Customer Segmentation (using K-Means clustering):

In this example, we'll use the popular K-Means clustering algorithm for customer segmentation. We'll use the scikit-learn library for Python.

**python**

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Load your customer data into a DataFrame

# Replace 'data.csv' with the path to your data file

data = pd.read\_csv('data.csv')

# Select relevant features for segmentation

X = data[['feature1', 'feature2', 'feature3']]

# Standardize the data to have mean=0 and variance=1

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Determine the optimal number of clusters (K) using the Elbow Method

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=42)

kmeans.fit(X\_scaled)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

plt.title('Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

# Based on the elbow method, choose an appropriate K (number of clusters)

# Apply K-Means clustering

k = 3 # You can choose the appropriate number of clusters

kmeans = KMeans(n\_clusters=k, init='k-means++', random\_state=42)

data['cluster'] = kmeans.fit\_predict(X\_scaled)

# Visualize the clusters

data['cluster'].value\_counts().plot(kind='bar')

plt.title('Cluster Distribution')

plt.xlabel('Cluster')

plt.ylabel('Count')

plt.show()

This code provides a basic template for customer segmentation using K-Means clustering. You'll need to customize it to your specific dataset and business requirements, including selecting the appropriate features, scaling, and choosing the number of clusters (K). Additionally, you can use more advanced techniques like hierarchical clustering, DBSCAN, or other machine learning algorithms for segmentation based on your data characteristics.

**APPLYING CLUSTERING ALGORITHMS:**

Customer segmentation using clustering algorithms is a common data science practice that helps businesses understand and group their customers based on shared characteristics or behavior. In this example, I'll show you how to perform customer segmentation using K-Means clustering, one of the most widely used clustering algorithms. We will use Python and the scikit-learn library for this purpose.

**Step 1: Data Preparation**

Load and prepare your customer data. Ensure that your dataset contains relevant features that can be used for segmentation. Common features may include demographic information, purchase history, and online behavior.

**python**

import pandas as pd

# Load your customer data into a DataFrame

# Replace 'data.csv' with the path to your data file

data = pd.read\_csv('data.csv')

# Select relevant features for segmentation

X = data[['feature1', 'feature2', 'feature3']]

**Step 2: Data Preprocessing**

Before applying K-Means clustering, it's important to preprocess the data by standardizing or normalizing it to ensure that all features have the same scale.

python

Copy code

from sklearn.preprocessing import StandardScaler

# Standardize the data

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

**Step 3: Determine the Optimal Number of Clusters**

You can use the Elbow Method to find the optimal number of clusters (K). This helps you decide how many segments to create.

python

Copy code

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=42)

kmeans.fit(X\_scaled)

wcss.append(kmeans.inertia\_)

# Plot the Within-Cluster-Sum-of-Squares (WCSS) for different values of K

plt.plot(range(1, 11), wcss)

plt.title('Elbow Method')

plt.xlabel('Number of clusters (K)')

plt.ylabel('WCSS')

plt.show()

Based on the "elbow" in the graph, you can select an appropriate number of clusters (K) for your segmentation.

**Step 4: Apply K-Means Clustering**

Now, apply K-Means clustering with the chosen number of clusters (K).

python

Copy code

k = 3 # Choose an appropriate number of clusters based on the Elbow Method

kmeans = KMeans(n\_clusters=k, init='k-means++', random\_state=42)

data['cluster'] = kmeans.fit\_predict(X\_scaled)

**Step 5: Visualize the Segments**

You can visualize the resulting customer segments to gain insights.

python

Copy code

import seaborn as sns

sns.scatterplot(data=data, x='feature1', y='feature2', hue='cluster', palette='viridis')

plt.title('Customer Segmentation')

plt.show()

This code provides a basic framework for customer segmentation using K-Means clustering. Make sure to adapt it to your specific dataset and business needs, including feature selection and fine-tuning the number of clusters. Additionally, you can explore other clustering algorithms like DBSCAN, hierarchical clustering, or Gaussian Mixture Models, depending on your data and objectives.

**DATA ANALYSIS:**

**VISUALIZATION:**

**Program:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

df = pd.read\_csv("D:/KV STUDY/tamil/Mall\_Customers.csv")

df

|  | **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 |
| **...** | ... | ... | ... | ... | ... |
| **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 |

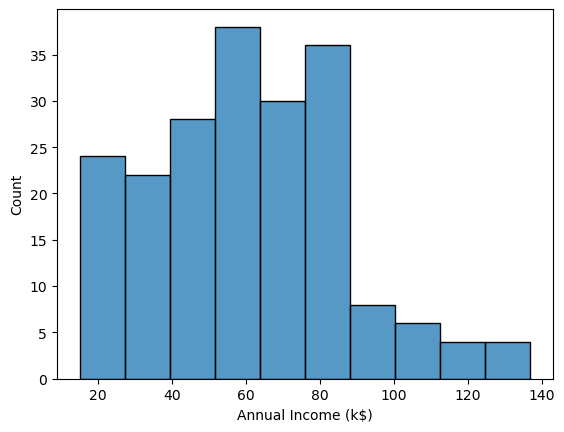
200 rows × 5 columns

df.describe()

|  | **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 |

sns.histplot(df['Annual Income (k$)'])

<Axes: xlabel='Annual Income (k$)', ylabel='Count'>



df.columns

Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',

'Spending Score (1-100)'],

dtype='object')

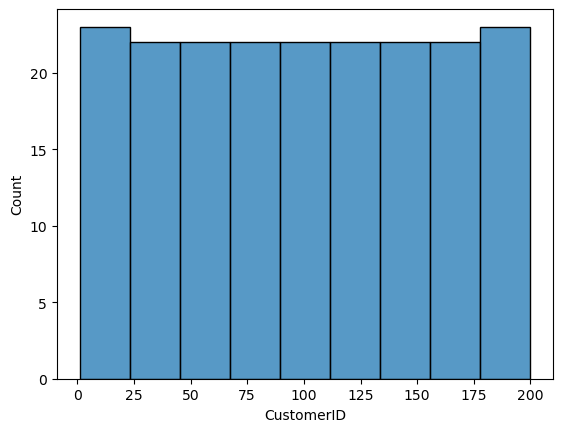
cols=['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',

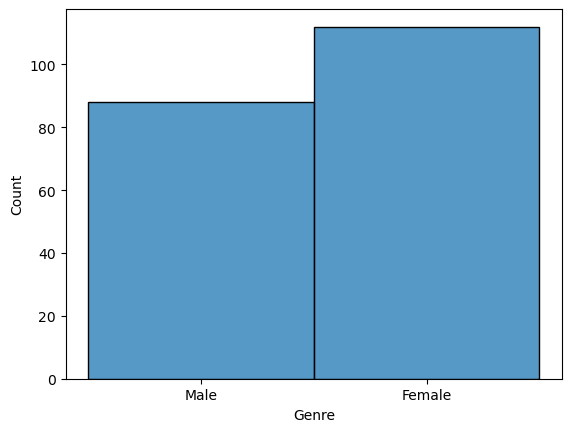
'Spending Score (1-100)']

for i in cols:

plt.figure()

sns.histplot(df[i])





A graph of age and age

Description automatically generated

A graph of a growing graph

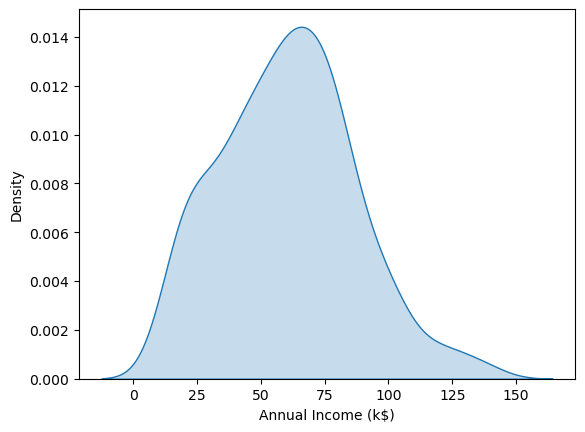
Description automatically generated with medium confidence

A graph of a number of blue bars

Description automatically generated with medium confidence

sns.kdeplot(df['Annual Income (k$)'], shade= True)

Axes: xlabel='Annual Income (k$)', ylabel='Density'>



import warnings

warnings.filterwarnings("ignore")

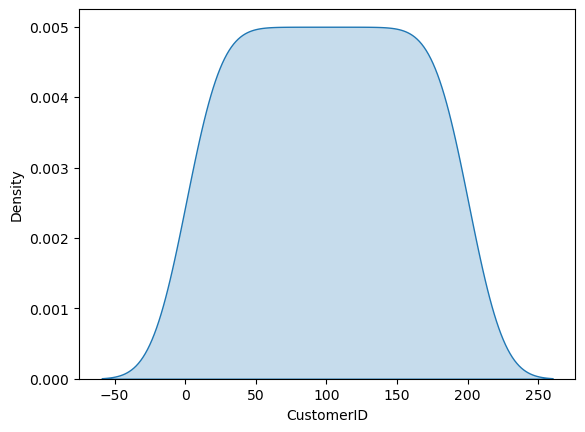
cols=['CustomerID', 'Age', 'Annual Income (k$)',

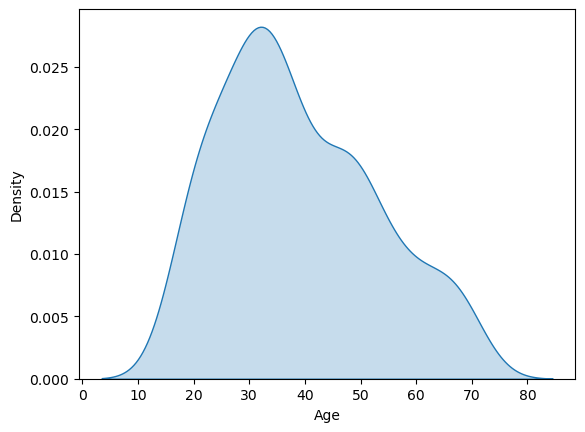
'Spending Score (1-100)']

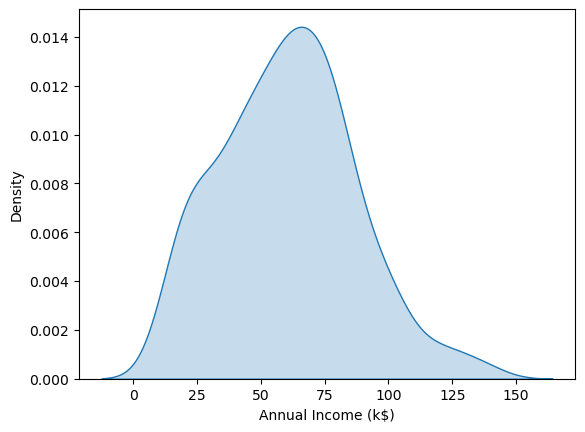
for x in cols:

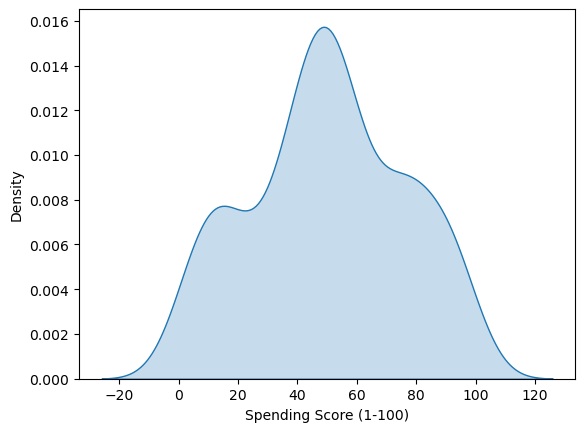
plt.figure()

sns.kdeplot(df[x],shade=True)









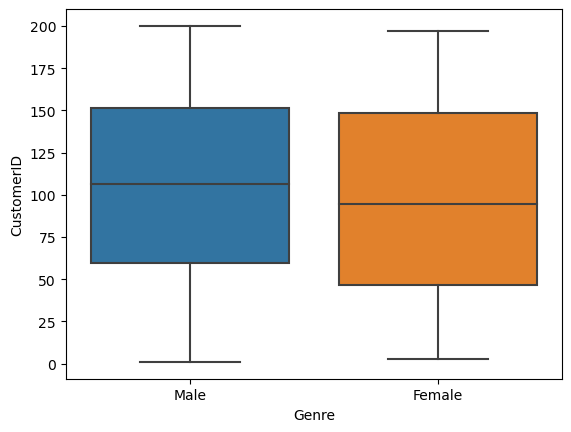
ols=['CustomerID', 'Age', 'Annual Income (k$)',

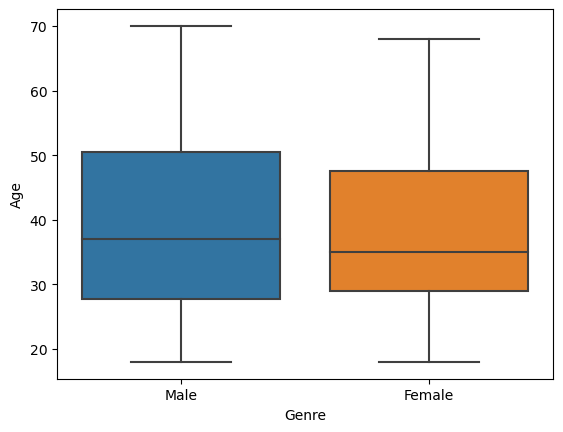
'Spending Score (1-100)']

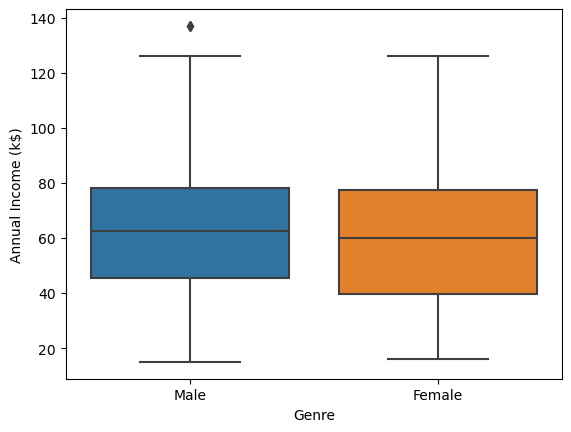
for i in cols:

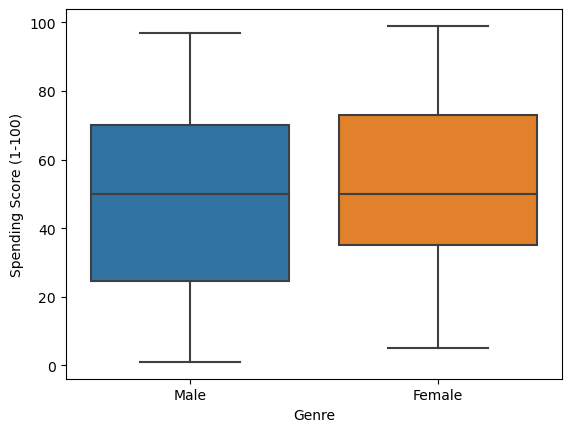
plt.figure()

sns.boxplot(data =df, x='Genre',y=df[i])









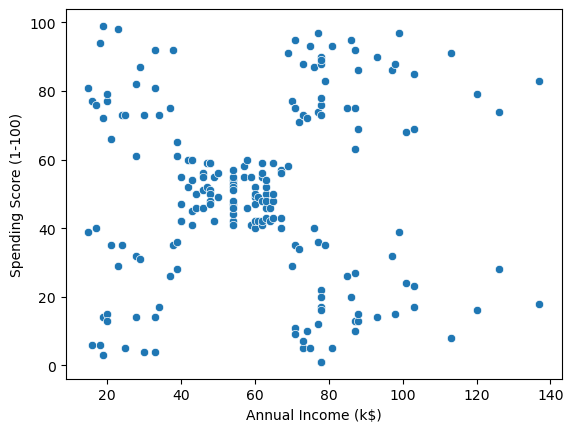
df['Genre'].value\_counts(normalize=True)

Female 0.56

Male 0.44

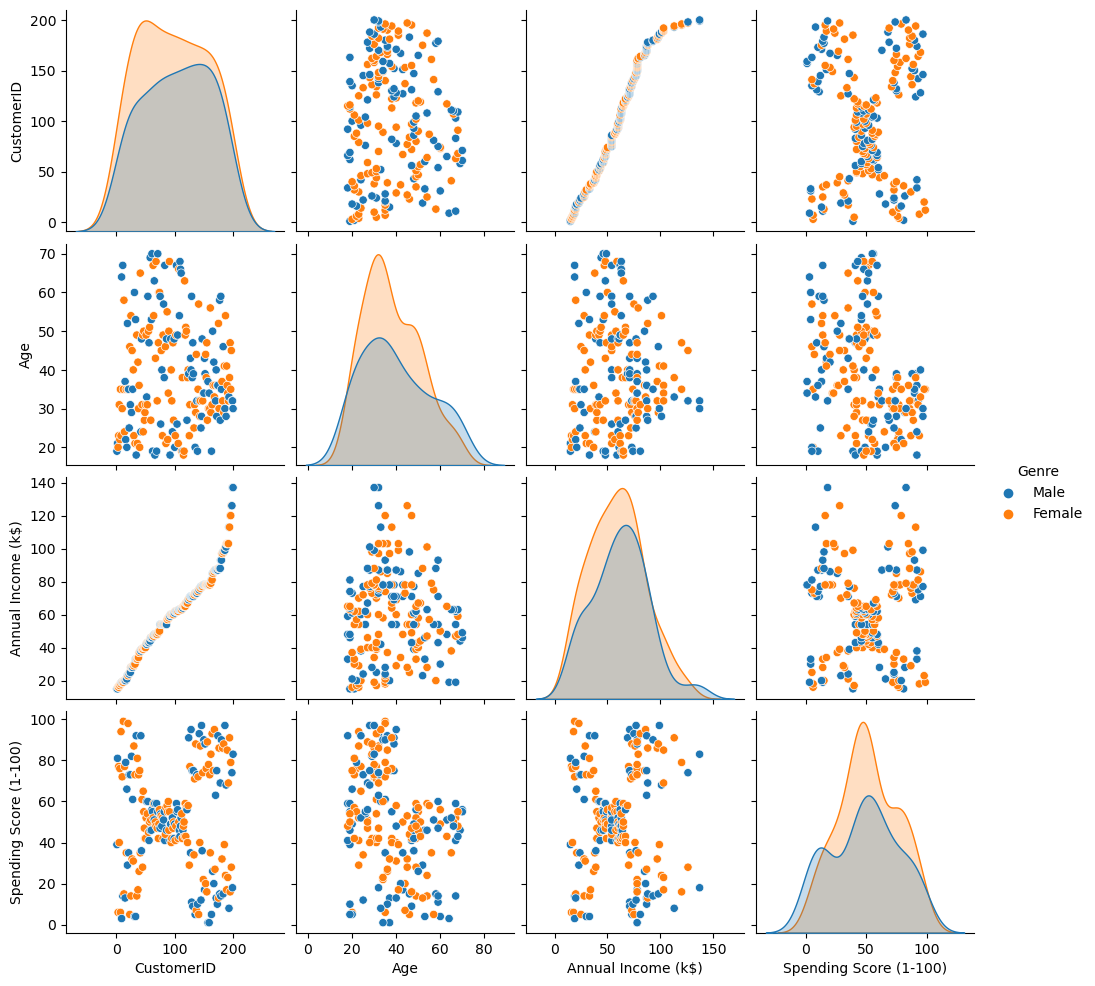
Name: Genre, dtype: float64

<Axes: xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>



sns.pairplot(df,hue**=**'Genre')

<seaborn.axisgrid.PairGrid at 0x1fd24481d50>



**def** g(df):

df2 **=** df.groupby(['Genre'])[['Annual Income (k$)','Age','Spending Score (1-100)']].mean()

**return** df2

​

​

df2 **=** g(df.copy())

print(df2)

Annual Income (k$) Age Spending Score (1-100)

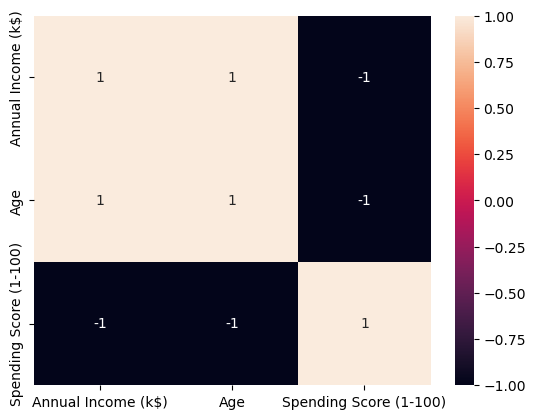
Genre

Female 59.250000 38.098214 51.526786

Male 62.227273 39.806818 48.511364

sns.heatmap(df2.corr(),annot**=True**)

<Axes: >



clustering1**=** KMeans(n\_clusters**=**4)

clustering1.fit(df[['Annual Income (k$)']])

KMeans

KMeans(n\_clusters=4)

clustering1.labels\_

array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,

3, 3])

df['income cluster']**=**clustering1.labels\_

df.head()

|  | **CustomerID** | **Genre** | **Age** | **Annual Income (k$)** | **Spending Score (1-100)** | **income cluster** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Male | 19 | 15 | 39 | 1 |
| **1** | 2 | Male | 21 | 15 | 81 | 1 |
| **2** | 3 | Female | 20 | 16 | 6 | 1 |
| **3** | 4 | Female | 23 | 16 | 77 | 1 |
| **4** | 5 | Female | 31 | 17 | 40 | 1 |

df['income cluster'].value\_counts()

2 68

0 62

1 50

3 20

Name: income cluster, dtype: int64

13278.112713472488

inertia\_scores **=** []

**for** i **in** range(1,11):

kmeans**=**KMeans(n\_clusters**=**i)

kmeans.fit(df[['Annual Income (k$)']])

inertia\_scores.append(kmeans.inertia\_)

inertia\_scores

[137277.28000000003,

48660.88888888888,

23517.330930930926,

13278.112713472488,

8481.496190476191,

5081.484660267269,

3949.275613275613,

2822.4996947496948,

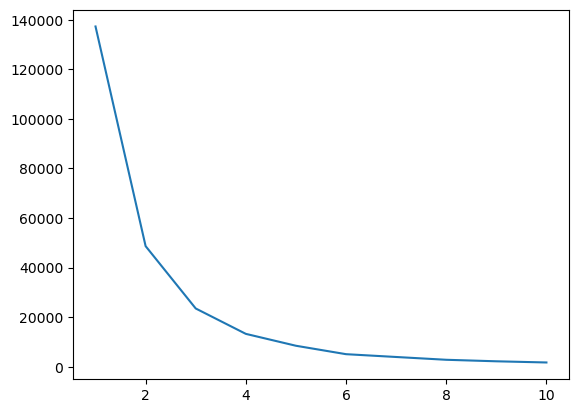
2211.8055555555557,

1734.167748917749]

plt.plot(range(1,11),inertia\_scores)

plt.show

<function matplotlib.pyplot.show(close=None, block=None)>



df.groupby('income cluster')[['Age', 'Spending Score (1-100)', 'Annual Income (k$)']].mean()

|  | **Age** | **Spending Score (1-100)** | **Annual Income (k$)** |
| --- | --- | --- | --- |
| **income cluster** |  |  |  |
| **0** | 36.838710 | 50.403226 | 77.806452 |
| **1** | 35.280000 | 49.480000 | 27.400000 |
| **2** | 43.970588 | 50.014706 | 54.764706 |
| **3** | 36.600000 | 52.000000 | 109.700000 |

clustering2**=**KMeans(n\_clusters**=**5)

clustering2.fit(df[['Annual Income (k$)','Spending Score (1-100)']])

KMeans

KMeans(n\_clusters=5)

clustering2.labels\_

df['spending score & annual income']**=**clustering2.labels\_

df.head()

|  | **CustomerID** | **Genre** | **Age** | **Annual Income (k$)** | **Spending Score (1-100)** | **income cluster** | **spending score & annual income** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Male | 19 | 15 | 39 | 1 | 3 |
| **1** | 2 | Male | 21 | 15 | 81 | 1 | 1 |
| **2** | 3 | Female | 20 | 16 | 6 | 1 | 3 |
| **3** | 4 | Female | 23 | 16 | 77 | 1 | 1 |
| **4** | 5 | Female | 31 | 17 | 40 | 1 | 3 |

clustering2.inertia\_

44448.45544793371

inertia\_scores2**=**[]

**for** i **in** range(1,11):

kmeans2**=**KMeans(n\_clusters**=**i)

kmeans2.fit(df[['Annual Income (k$)','Spending Score (1-100)']])

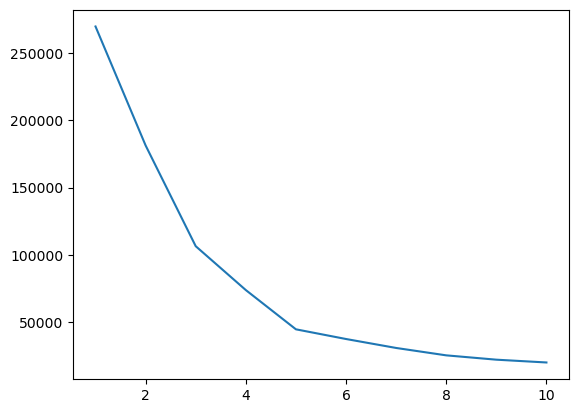
inertia\_scores2.append(kmeans2.inertia\_)

plt.plot(range(1,11),inertia\_scores2)

[<matplotlib.lines.Line2D at 0x1fd258e3050>]

enters**=**pd.DataFrame(clustering2.cluster\_centers\_)

centers.columns**=**['x','y']

centers

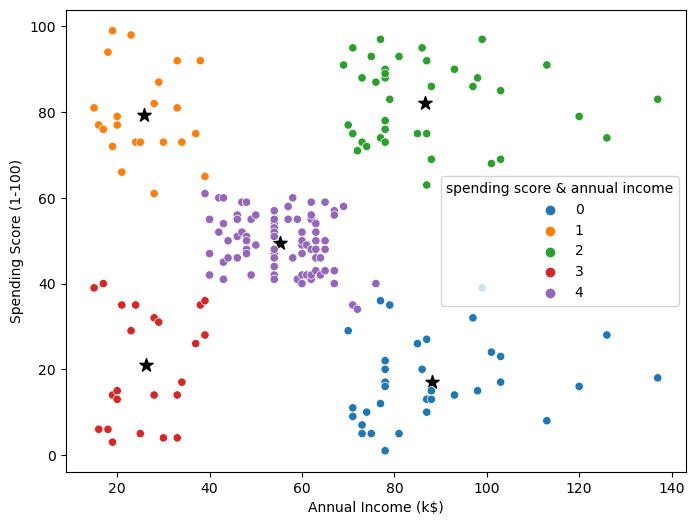
|  | **x** | **y** |
| --- | --- | --- |
| **0** | 88.200000 | 17.114286 |
| **1** | 25.727273 | 79.363636 |
| **2** | 86.538462 | 82.128205 |
| **3** | 26.304348 | 20.913043 |
| **4** | 55.296296 | 49.518519 |

plt.figure(figsize**=**(8

plt.scatter(x**=**centers['x'],y**=**centers['y'],s**=**100,c**=**'black',marker**=**'\*')

sns.scatterplot(data **=** df,x**=**'Annual Income (k$)',y**=** 'Spending Score (1-100)',hue**=**'spending score & annual income',palette**=**'tab10')

plt.savefig('Clustering Bivariate.png')



**INTERPRETATION:**

Customer segmentation is not complete without interpretation, as the goal is to derive actionable insights from the segmentation results. Once you've performed customer segmentation using data science techniques (such as clustering), here's how you can interpret the segments:

**Step 1: Review the Segment Characteristics**

**Descriptive Statistics:**

Calculate and compare the descriptive statistics for each segment. This includes means, standard deviations, and other relevant statistics for the features used in the segmentation.

**Visualization:**

Create visualizations that show the feature distributions within each segment. Box plots, histograms, or bar charts can help you understand the differences between segments.

**python**

import matplotlib.pyplot as plt

import seaborn as sns

# Example: Plot histograms of a feature for each segment

sns.set\_style('whitegrid')

plt.figure(figsize=(10, 6))

sns.histplot(data=data, x='feature1', hue='cluster', kde=True, common\_norm=False, palette='viridis')

plt.title('Distribution of Feature 1 for Each Segment')

plt.xlabel('Feature 1')

plt.show()

**Step 2: Investigate Segment Characteristics**

**Feature Importance:**

Use feature importance analysis to identify which features contributed most to the segmentation. This can be done with methods like examining the centroid values for K-Means clusters or analyzing decision trees for hierarchical clustering.

python

Copy code

# If using K-Means, you can examine the cluster centers (centroids)

cluster\_centers = scaler.inverse\_transform(kmeans.cluster\_centers\_)

cluster\_centers\_df = pd.DataFrame(cluster\_centers, columns=X.columns)

print(cluster\_centers\_df)

Segment Descriptions: Write descriptions or summaries of each segment based on their characteristics. For example, "Segment 1 represents high-income, tech-savvy customers who frequently make online purchases."

**Step 3: Business Insights**

**Marketing Strategies:**

Develop specific marketing strategies for each segment based on their unique characteristics. For example, offer technology-related products to tech-savvy segments and promote loyalty programs to high-value segments.

**Product Recommendations**:

Tailor product recommendations for each segment. For instance, recommend premium products to high-income segments and budget options to price-conscious segments.

**Customer Engagement:**

Craft communication and engagement strategies based on the preferred channels and behavior of each segment. High social media engagement segments might benefit from influencer marketing.

**Pricing Strategies:**

Adjust pricing strategies according to the price sensitivity of each segment. Offer discounts to price-sensitive segments while maintaining premium pricing for high-value segments.

**Customer Retention**:

Identify segments that are at risk of churning and implement customer retention initiatives specifically targeted at those segments.

**Step 4: Monitor and Iterate**

Continuously monitor the performance of your strategies for each segment. Use feedback and data from ongoing campaigns to refine your customer segmentation and make improvements to your strategies. Customer segmentation is an ongoing process, and your segments may evolve over time.

Interpreting customer segments is crucial for making data-driven decisions that lead to improved customer experiences and business outcomes. It enables you to personalize marketing efforts, enhance customer engagement, and optimize business operations.

**CONCLUSION**

Customer segmentation is performed on the company's customers data and with the help of K-means clustering machine learning algorithm customers are divided using features like total spending and annual income, this study also proves that the dividing customers on the basis of behavioral characteristics is a better solution for existing customer segmentation problem and K-means clustering algorithm is identified as a good choice for this approach.