BIA Project

Group Members:

Muhammad Hammad 01-135211-102.

Hammad-ul-Hasan 01-135211-058.

**Customer Segmentation:**

**1. High Spending, Young Age:** Young customers (20-35 years old) with a high spending score (70-100), indicating impulsive or trend-driven buyers.

**2. Middle Spending,** Middle Age: Middle-aged customers (36-50) with a medium spending score (40-69), likely balancing needs and budgets.

**3. Low Spend, Older:** Older customers (51+) with a low spend score (0-39), focusing on needs and limited spending.

**Logistic regression:**

**Code:**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix**

**# Load the dataset from a CSV file**

**data = pd.read\_csv('Mall\_Custom.csv')**

**# Print the column names to verify them**

**print(data.columns)**

**# Feature selection: Select relevant columns (excluding CustomerID, as it's not predictive)**

**X = data[['Age', 'Annual Income', 'Spending Score']] # Features**

**y = data['Genre'] # Target (Assuming Genre is the column to predict)**

**# Convert categorical target (Genre) to numerical if needed**

**y = y.map({'Male': 0, 'Female': 1}) # Example mapping (adjust as per your dataset)**

**# Split the dataset into training and testing sets (80% train, 20% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Train a Logistic Regression model**

**model = LogisticRegression()**

**model.fit(X\_train, y\_train) # Fit the model to the training data**

**# Make predictions on the test data**

**y\_pred = model.predict(X\_test)**

**# Calculate performance metrics**

**accuracy = accuracy\_score(y\_test, y\_pred) # Overall correctness**

**precision = precision\_score(y\_test, y\_pred) # True positives / predicted positives**

**recall = recall\_score(y\_test, y\_pred) # True positives / actual positives**

**f1 = f1\_score(y\_test, y\_pred) # Harmonic mean of precision and recall**

**# Print the evaluation results**

**print(f"Accuracy: {accuracy:.2f}")**

**print(f"Precision: {precision:.2f}")**

**print(f"Recall: {recall:.2f}")**

**print(f"F1 Score: {f1:.2f}")**

**# Confusion matrix plot**

**cm = confusion\_matrix(y\_test, y\_pred)**

**plt.figure(figsize=(6, 5)) # Set plot size**

**sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Male', 'Female'], yticklabels=['Male', 'Female'])**

**plt.title('Confusion Matrix')**

**plt.xlabel('Predicted')**

**plt.ylabel('Actual')**

**plt.show()**

**Decision Tree:**

**Code:**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score**

**# Load the dataset from a CSV file**

**data = pd.read\_csv('Mall\_Custom.csv')**

**# Print the column names to verify them**

**print(data.columns)**

**# Feature selection: Select relevant columns (excluding CustomerID, as it's not predictive)**

**X = data[['Age', 'Annual Income', 'Spending Score']] # Features**

**y = data['Genre'] # Target (Assuming Genre is the column to predict)**

**# Convert categorical target (Genre) to numerical if needed**

**y = y.map({'Male': 0, 'Female': 1}) # Example mapping (adjust as per your dataset)**

**# Split the dataset into training and testing sets (80% train, 20% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Train a Decision Tree model**

**model = DecisionTreeClassifier(random\_state=42)**

**model.fit(X\_train, y\_train) # Fit the model to the training data**

**# Make predictions on the test data**

**y\_pred = model.predict(X\_test)**

**# Calculate performance metrics**

**accuracy = accuracy\_score(y\_test, y\_pred) # Overall correctness**

**precision = precision\_score(y\_test, y\_pred) # True positives / predicted positives**

**recall = recall\_score(y\_test, y\_pred) # True positives / actual positives**

**f1 = f1\_score(y\_test, y\_pred) # Harmonic mean of precision and recall**

**# Print the evaluation results**

**print(f"Accuracy: {accuracy:.2f}")**

**print(f"Precision: {precision:.2f}")**

**print(f"Recall: {recall:.2f}")**

**print(f"F1 Score: {f1:.2f}")**

**# Create a bar plot for the performance metrics**

**metrics = ['Accuracy', 'Precision', 'Recall', 'F1 Score']**

**scores = [accuracy, precision, recall, f1]**

**plt.figure(figsize=(8, 6))**

**sns.barplot(x=metrics, y=scores, palette='viridis')**

**plt.title('Performance Metrics of Decision Tree Model')**

**plt.xlabel('Metrics')**

**plt.ylabel('Scores')**

**plt.ylim(0, 1) # Set y-axis limits from 0 to 1**

**plt.show()**

**Random Forest:**

**Code:**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score**

**# Load the dataset from a CSV file**

**data = pd.read\_csv('Mall\_Custom.csv')**

**# Print the column names to verify them**

**print(data.columns)**

**# Feature selection: Select relevant columns (excluding CustomerID, as it's not predictive)**

**X = data[['Age', 'Annual Income', 'Spending Score']] # Features**

**y = data['Genre'] # Target (Assuming Genre is the column to predict)**

**# Convert categorical target (Genre) to numerical if needed**

**y = y.map({'Male': 0, 'Female': 1}) # Example mapping (adjust as per your dataset)**

**# Split the dataset into training and testing sets (80% train, 20% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Train a Decision Tree model**

**model = DecisionTreeClassifier(random\_state=42)**

**model.fit(X\_train, y\_train) # Fit the model to the training data**

**# Make predictions on the test data**

**y\_pred = model.predict(X\_test)**

**# Calculate performance metrics**

**accuracy = accuracy\_score(y\_test, y\_pred) # Overall correctness**

**precision = precision\_score(y\_test, y\_pred) # True positives / predicted positives**

**recall = recall\_score(y\_test, y\_pred) # True positives / actual positives**

**f1 = f1\_score(y\_test, y\_pred) # Harmonic mean of precision and recall**

**# Print the evaluation results**

**print(f"Accuracy: {accuracy:.2f}")**

**print(f"Precision: {precision:.2f}")**

**print(f"Recall: {recall:.2f}")**

**print(f"F1 Score: {f1:.2f}")**

**# Visualization of Age vs Spending Score**

**plt.figure(figsize=(8, 6))**

**sns.scatterplot(x=data['Age'], y=data['Spending Score'], hue=data['Genre'], palette='viridis', s=100)**

**plt.title('Age vs Spending Score')**

**plt.xlabel('Age')**

**plt.ylabel('Spending Score (1-100)')**

**plt.legend(title='Genre')**

**plt.show()**

**Support Vector Machine:**

**Code:**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.svm import SVC**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score**

**# Load the dataset from a CSV file**

**data = pd.read\_csv('Mall\_Custom.csv')**

**# Print the column names to verify them**

**print(data.columns)**

**# Feature selection: Select relevant columns (excluding CustomerID, as it's not predictive)**

**X = data[['Age', 'Spending Score']] # Features**

**y = data['Genre'] # Target (Assuming Genre is the column to predict)**

**# Convert categorical target (Genre) to numerical if needed**

**y = y.map({'Male': 0, 'Female': 1}) # Example mapping (adjust as per your dataset)**

**# Split the dataset into training and testing sets (80% train, 20% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Train a Support Vector Machine (SVM) model**

**model = SVC(kernel='linear') # Linear Kernel for simplicity**

**model.fit(X\_train, y\_train) # Fit the model to the training data**

**# Make predictions on the test data**

**y\_pred = model.predict(X\_test)**

**# Calculate performance metrics**

**accuracy = accuracy\_score(y\_test, y\_pred) # Overall correctness**

**precision = precision\_score(y\_test, y\_pred) # True positives / predicted positives**

**recall = recall\_score(y\_test, y\_pred) # True positives / actual positives**

**f1 = f1\_score(y\_test, y\_pred) # Harmonic mean of precision and recall**

**# Print the evaluation results**

**print(f"Accuracy: {accuracy:.2f}")**

**print(f"Precision: {precision:.2f}")**

**print(f"Recall: {recall:.2f}")**

**print(f"F1 Score: {f1:.2f}")**

**# Visualize the data with a scatter plot (Age vs Spending Score)**

**plt.figure(figsize=(8, 6))**

**sns.scatterplot(x='Age', y='Spending Score', hue='Genre', data=data, palette='viridis')**

**plt.title('Scatter Plot of Age vs Spending Score')**

**plt.xlabel('Age')**

**plt.ylabel('Spending Score (1-100)')**

**plt.show()**