Title: Predicting Buyer Type using Random Forest Classifier
Name: Aanjneya Nayak
Roll No.: 20240110030001
Course: CSE - AI
Project Type: Machine Learning Classification Project
Date: 22, April 2025

Introduction

In today's e-commerce and retail environment, understanding buyer behavior is crucial for enhancing user experience and increasing sales. This project aims to build a machine learning model that predicts whether a customer is a **bargain hunter** or a **premium buyer** based on their spending habits.

The dataset contains the following columns:

- total_spent: Total money spent by the customer.
- avg_purchase_value: Average amount spent per purchase.
- visits_per_month: Number of times the customer visits per month.
- buyer_type: The target variable (bargain_hunter or premium_buyer).

We used a **Random Forest Classifier** for this classification task due to its accuracy and robustness in handling tabular data.

Methodology

Step 1: Data Preprocessing

- Loaded the CSV file using pandas.
- Checked for missing values and cleaned the data.
- Encoded the target column buyer_type using label encoding:
    - bargain_hunter → 0
    - premium_buyer → 1

Step 2: Feature Selection

- Selected total_spent, avg_purchase_value, and visits_per_month as input features (X).
- Used buyer_type as the output label (y).

Step 3: Model Training

- Split the data into **training (80%)** and **testing (20%)** sets using train_test_split.
- Trained a **RandomForestClassifier** on the training data.

Step 4: Evaluation

- Evaluated performance using:
    - **Confusion Matrix**
    - **Accuracy**
    - **Precision**
    - **Recall**
- Visualized the confusion matrix using seaborn.heatmap.

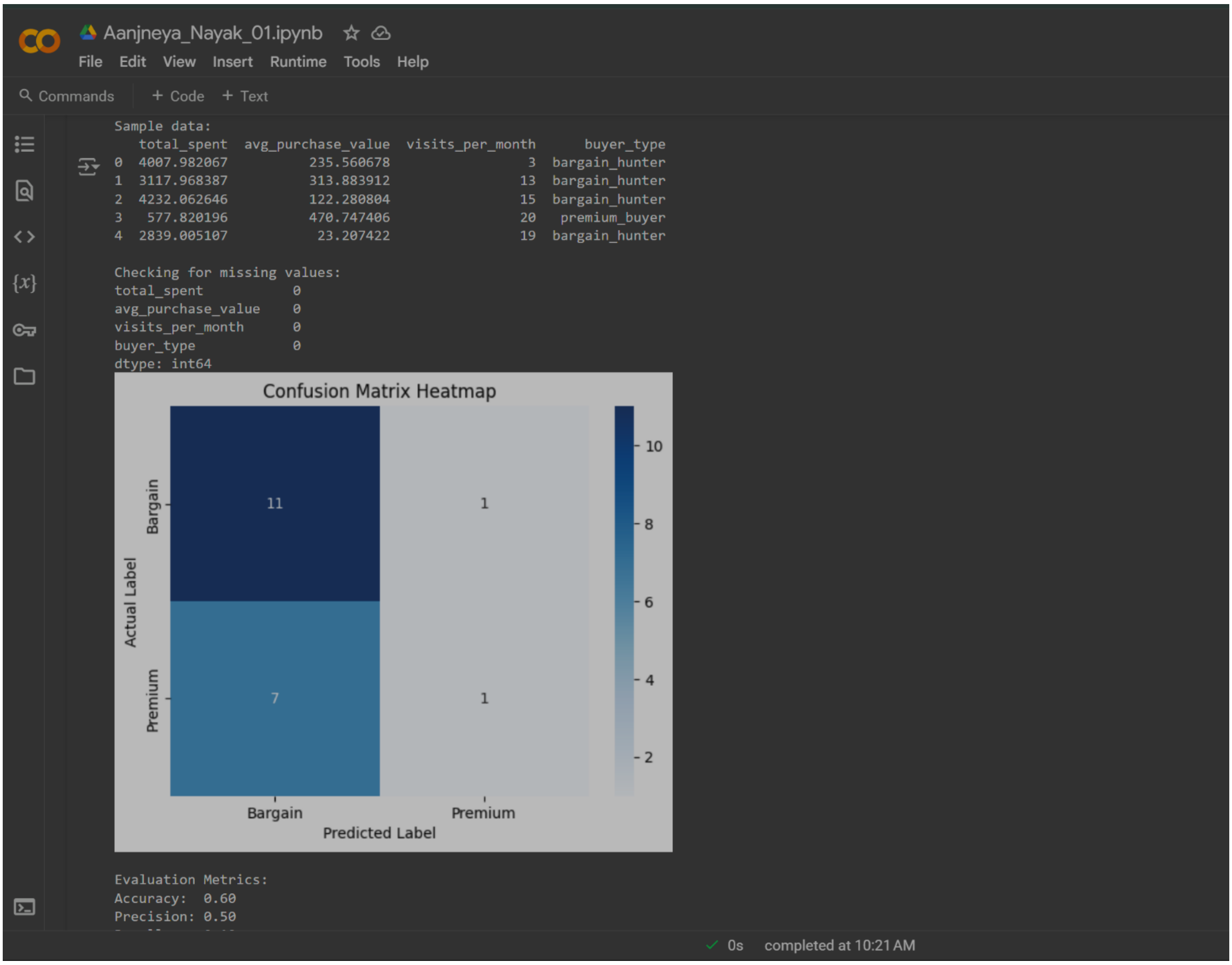
CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('/content/customer_behavior.csv')

# View data
print("Sample data:")
print(df.head())

# Check for missing values
print("\nChecking for missing values:")
print(df.isnull().sum())

# Encode the buyer_type column
df['buyer_type'] = df['buyer_type'].map({'bargain_hunter': 0, 'premium_buyer': 1})

# Features and labels
X = df[['total_spent', 'avg_purchase_value', 'visits_per_month']]
y = df['buyer_type']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Model training
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Bargain', 'Premium'], yticklabels=['Bargain', 'Premium'])
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.title('Confusion Matrix Heatmap')
plt.show()

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("\nEvaluation Metrics:")
print(f"Accuracy:  {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall:    {recall:.2f}")
```

Output/Result

Sample Output Screenshot:



```
Sample data:
   total_spent  avg_purchase_value  visits_per_month      buyer_type
0  4007.982067          235.560678                 3  bargain_hunter
1  3117.968387          313.883912                13  bargain_hunter
2  4232.062646          122.280804                15  bargain_hunter
3   577.820196          470.747406                20   premium_buyer
4  2839.005107           23.207422                19  bargain_hunter

Checking for missing values:
total_spent           0
avg_purchase_value    0
visits_per_month      0
buyer_type            0
dtype: int64
```

```
Evaluation Metrics:
Accuracy:  0.60
Precision: 0.50
```

References/Credits

- Dataset created manually based on hypothetical customer behavior.
- Scikit-learn documentation
- Pandas documentation
- Seaborn documentation