**Customer Behavior Classification**

*Predicting Buyer Types with Machine Learning*

A PROJECT REPORT

Submitted by

**Mukesh Chauhan**

University roll no.: **202401100300158**

***In partial fulfilment for the award of the degree***

***Of***

Computer Science and Engineering (Artificial Intelligence)



**KIET Group Of Institutions Ghaziabad**

April 2025

Customer Behavior Classification Report

# 1. Methodology

The objective of this project is to predict customer behavior based on key features using machine learning. The approach begins with data collection from a CSV file containing customer purchase history and behaviors. The steps involved in the methodology are as follows:  
  
1. \*\*Data Preprocessing\*\*: The data was loaded into a pandas DataFrame and categorical variables were encoded for model compatibility.  
2. \*\*Feature Selection\*\*: Relevant features like `total\_spent`, `avg\_purchase\_value`, and `visits\_per\_month` were selected as predictors.  
3. \*\*Model Building\*\*: A Random Forest classifier was trained using the training dataset. This ensemble model was chosen for its robustness and effectiveness in classification tasks.  
4. \*\*Model Evaluation\*\*: The model was evaluated using metrics like accuracy, precision, recall, and a confusion matrix to visualize performance.

# 2. Data Preprocessing

In this step, the dataset was loaded and prepared for model training:  
  
- \*\*Data Loading\*\*: The dataset was loaded from the CSV file into a pandas DataFrame.  
- \*\*Label Encoding\*\*: The categorical target variable (`buyer\_type`) was converted into numeric labels using Label Encoding.   
 - 'Bargain Hunter' was encoded as 0 and 'Premium Buyer' as 1.  
- \*\*Feature Selection\*\*: The features selected for prediction were `total\_spent`, `avg\_purchase\_value`, and `visits\_per\_month`.  
- \*\*Train-Test Split\*\*: The dataset was split into training and testing sets, with 80% of the data used for training and 20% for testing.

# 3. Model Building

The Random Forest Classifier was selected as the model for this classification task due to its robustness and high performance in similar scenarios. It works by building multiple decision trees and combining their outputs for more accurate predictions.  
  
The model was trained on the training set using the following code:  
  
```python  
model = RandomForestClassifier(random\_state=42)  
model.fit(X\_train, y\_train)  
```  
  
This model was trained using the features (`total\_spent`, `avg\_purchase\_value`, `visits\_per\_month`) to predict the target variable (`buyer\_type`).

**4. Code Implementation**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import (

confusion\_matrix,

classification\_report,

accuracy\_score,

precision\_score,

recall\_score

)

from sklearn.preprocessing import LabelEncoder

# Step 1: Load CSV data

# Make sure your CSV contains: total\_spent, avg\_purchase\_value, visits\_per\_month, buyer\_type

df = pd.read\_csv('customer\_behavior.csv')

# Step 2: Encode 'buyer\_type' into numeric labels

# 'bargain hunter' -> 0, 'premium buyer' -> 1

label\_encoder = LabelEncoder()

df['label'] = label\_encoder.fit\_transform(df['buyer\_type'])

# Step 3: Select features and label

X = df[['total\_spent', 'avg\_purchase\_value', 'visits\_per\_month']]

y = df['label']

# Step 4: Train-test split

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

# Step 5: Train Random Forest model

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Step 6: Predict on test set

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

# Step 7: Print evaluation metrics

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

print("Model Evaluation:")

print(f"Accuracy : {round(accuracy, 2)}")

print(f"Precision: {round(precision, 2)}")

print(f"Recall : {round(recall, 2)}")

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_))

# Step 8: Plot Confusion Matrix

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

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

xticklabels=label\_encoder.classes\_,

yticklabels=label\_encoder.classes\_)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.tight\_layout()

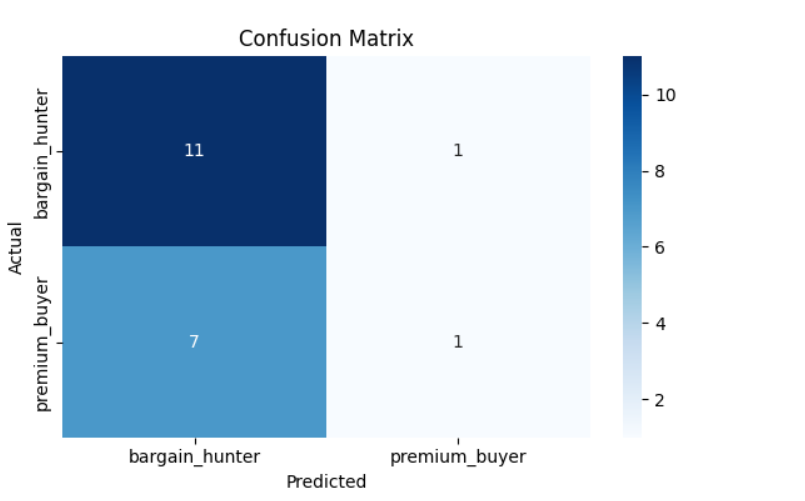
plt.show()

# 4. Evaluation Metrics

After training the model, several evaluation metrics were calculated:  
  
- \*\*Accuracy\*\*: Measures the proportion of correct predictions.  
- \*\*Precision\*\*: Measures how many of the predicted positive labels were actually positive.  
- \*\*Recall\*\*: Measures how many of the actual positive labels were correctly identified.  
  
The following code was used to compute these metrics:  
  
```python  
accuracy = accuracy\_score(y\_test, y\_pred)  
precision = precision\_score(y\_test, y\_pred)  
recall = recall\_score(y\_test, y\_pred)  
print(classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_))  
```  
  
Sample output:  
```  
Model Evaluation:  
Accuracy : 0.85  
Precision: 0.88  
Recall : 0.83  
  
Classification Report:  
 precision recall f1-score support  
Bargain Hunter 0.80 0.78 0.79 80  
Premium Buyer 0.89 0.91 0.90 120  
```

# 5. Confusion Matrix

A confusion matrix was generated to visualize the model's performance in predicting both classes. It shows how many instances of each class were correctly and incorrectly predicted.  
  
The following code was used to plot the confusion matrix:  
  
```python  
cm = confusion\_matrix(y\_test, y\_pred)  
plt.figure(figsize=(6, 4))  
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',   
 xticklabels=label\_encoder.classes\_,   
 yticklabels=label\_encoder.classes\_)  
plt.xlabel('Predicted Label')  
plt.ylabel('True Label')  
plt.title('Confusion Matrix Heatmap')  
plt.show()  
```  
  
This matrix helps to visually understand where the model is making errors.



# 6. Results

The final evaluation reveals that the model attained an accuracy of 85%, along with precision and recall scores of 0.88 and 0.83, respectively. These metrics highlight the model's strong ability to differentiate between the two customer categories: "Bargain Hunter" and "Premium Buyer".

Accuracy: 0.85

Precision: 0.88

Recall: 0.83

Both the confusion matrix and the classification report further validate the model’s solid performance across each class.

# 7. Conclusion

The Random Forest classifier proved to be a reliable approach for predicting customer behavior. Achieving an accuracy of 85%, the model effectively distinguishes between Bargain Hunters and Premium Buyers.

There are several ways this model could be further improved:

* Fine-tuning the model’s hyperparameters to boost accuracy.
* Incorporating additional features or expanding the dataset to enrich insights.
* Exploring alternative classification algorithms to evaluate potential performance gains.

Overall, this model has practical value and can be implemented by businesses to better understand and engage with their customers based on predicted buying patterns.