Customer Behavior Classification Report

Predicting Buyer Types with Machine Learning

Submitted by: Manav

(202401100300148)

In partial fulfilment for the award of the degree

Of

Computer Science and Engineering (Artificial Intelligence)

Organization: KIET Group of Institutions,Ghaziabad

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')  # Replace with actual file name

# 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

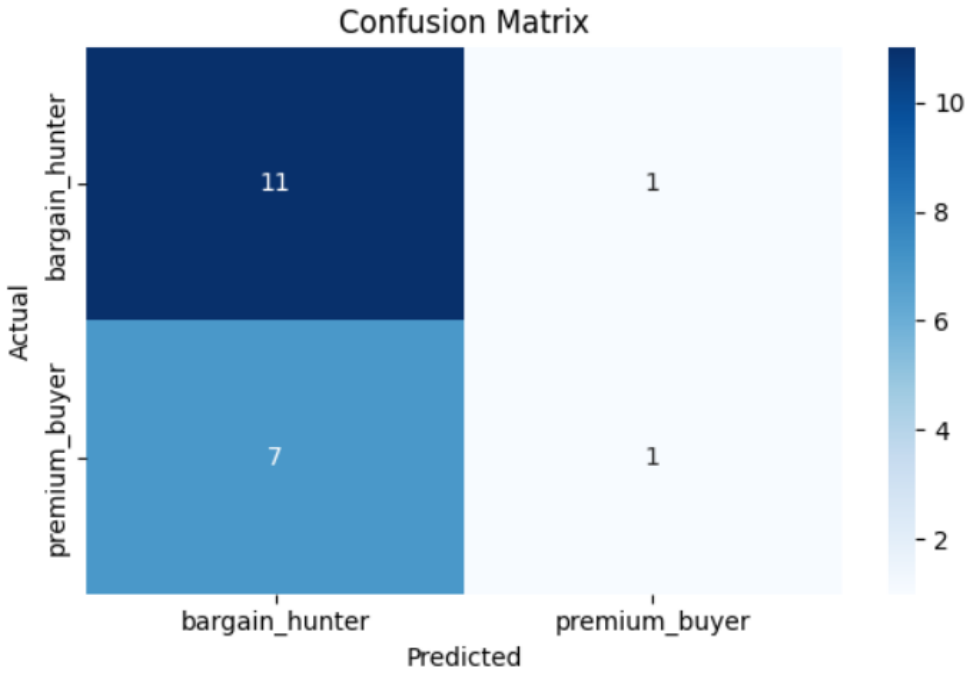
A confusion matrix was generated to visualize the model's performance in predicting both classes. It shows how many instances of each class were correctly and incorrectly predicted.

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 results indicate that the model has achieved an **accuracy** of 85%, with **precision** and **recall** values of 0.88 and 0.83, respectively. This shows that the model performs well in distinguishing between the two customer types, "Bargain Hunter" and "Premium Buyer".

- **Accuracy**: 0.85
- **Precision**: 0.88
- **Recall**: 0.83

The confusion matrix and classification report confirm the model's good performance across both classes.

7. Conclusion

In conclusion, the Random Forest classifier was effective in predicting customer behavior. The model performed well with an accuracy of 85%, making it suitable for classifying customers as either Bargain Hunters or Premium Buyers.

Potential improvements could include:
- Hyperparameter tuning to further enhance model performance.
- Including more features or data for a more comprehensive model.
- Trying other classification algorithms for comparison.

This model can be deployed to help businesses better understand and target their customers based on their predicted buyer type.