Predict Product Return

A PROJECT REPORT

Submitted by

Raj Ojha

University roll no: 202401100300192

In partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering - Artificial Intelligence

KIET Groups of Institutions

Ghaziabad

INTRODUCTION

In the rapidly evolving world of e-commerce, managing product returns is a crucial aspect of improving customer satisfaction and optimizing business operations. The ability to predict whether a product will be returned can help businesses make data-driven decisions, reduce operational costs, and enhance inventory management.

This report focuses on the development of a predictive model aimed at forecasting product returns based on key features, such as purchase amount, review score, and delivery time. By leveraging historical transaction data, the model employs machine learning techniques to predict whether a customer is likely to return a product.

A Random Forest Classifier has been implemented, with hyperparameter tuning via GridSearchCV to ensure optimal performance. The model is evaluated using various metrics, including accuracy, classification report, and confusion matrix, and is accompanied by visualizations to provide deeper insights into the underlying patterns influencing return behavior.

The main objective of this predictive model is to provide valuable insights for e-commerce businesses to proactively address potential product returns, optimize customer service, and improve the overall shopping experience. Through the predictions generated, businesses can better anticipate return rates and make strategic decisions that contribute to higher efficiency and profitability.

METHODOLOGY

The approach for predicting product returns involves the following key steps:

1. **Data Collection and Preprocessing**
   The dataset used in this model contains transaction information, including purchase amount, review score, days to delivery, and whether the product was returned. Missing values are handled by dropping rows with incomplete data in critical columns such as `purchase_amount`, `review_score`, and `days_to_delivery`. Additionally, the target variable `returned` is converted to a binary format, where 'yes' is mapped to 1 and 'no' to 0.

2. **Feature Selection**
   The model uses three key features:

   - `purchase_amount`: The total value of the product purchased.
   - `review_score`: The rating given by the customer (from 0 to 5).
   - `days_to_delivery`: The number of days it took for the product to be delivered.
     These features are selected based on their potential influence on the likelihood of a product being returned.

3. **Model Training**
   A Random Forest Classifier is chosen as the model due to its ability to handle non-linear relationships and its robustness to overfitting. The model is trained on the training dataset (80% of the data), and hyperparameter tuning is performed using GridSearchCV to find the optimal model parameters.

4. **Model Evaluation**
   After training, the model is tested on the remaining 20% of the data (test set). Evaluation metrics, such as accuracy, confusion matrix, and classification report, are used to assess the performance of the model.

5. **User Prediction**
   The trained model is used to predict product returns based on user input, including purchase amount, review score, and delivery time. The model outputs whether the product is likely to be returned or not.

6. **Visualization**
   Several visualizations are created to better understand the data distribution and model performance, including:

   - **Purchase Amount Distribution**: Showing how the purchase amount correlates with return behavior.

   - **Review Score vs Return**: Illustrating how customer ratings relate to the likelihood of returns.

   - **Delivery Days vs Return**: Analyzing the effect of delivery time on return rates.

   - **Confusion Matrix**: Evaluating the true vs predicted return outcomes.

This methodology ensures a robust model that can accurately predict product returns and provide actionable insights for businesses.

CODE

```
#Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

#Load and preprocess data
def load_data():
    df = pd.read_csv("/content/product_return.csv")
    df = df.dropna(subset=['purchase_amount', 'review_score', 'days_to_delivery', 'returned'])
    df['returned'] = df['returned'].map({'yes': 1, 'no': 0})
    return df

df = load_data()

# Split data
X = df[['purchase_amount', 'review_score', 'days_to_delivery']]
y = df['returned']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Train model with class weights and hyperparameter tuning
model = RandomForestClassifier(random_state=42, class_weight='balanced')

#Hyperparameter tuning with GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, None],
    'min_samples_split': [2, 5, 10],
```

```
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_jobs=-1,
scoring='accuracy')
grid_search.fit(X_train, y_train)

#Best model from grid search
best_model = grid_search.best_estimator_

#Predict using the best model
y_pred = best_model.predict(X_test)

#User input section
print(" Product Return Prediction App")

purchase_amount = float(input("Enter Purchase Amount: "))
review_score = float(input("Enter Review Score (0 to 5): "))
days_to_delivery = int(input("Enter Days to Delivery: "))

#Predict based on user input
user_input = pd.DataFrame({
    'purchase_amount': [purchase_amount],
    'review_score': [review_score],
    'days_to_delivery': [days_to_delivery]
})
prediction = best_model.predict(user_input)[0]

if prediction == 1:
    print(" The product **will be returned**.")
else:
```

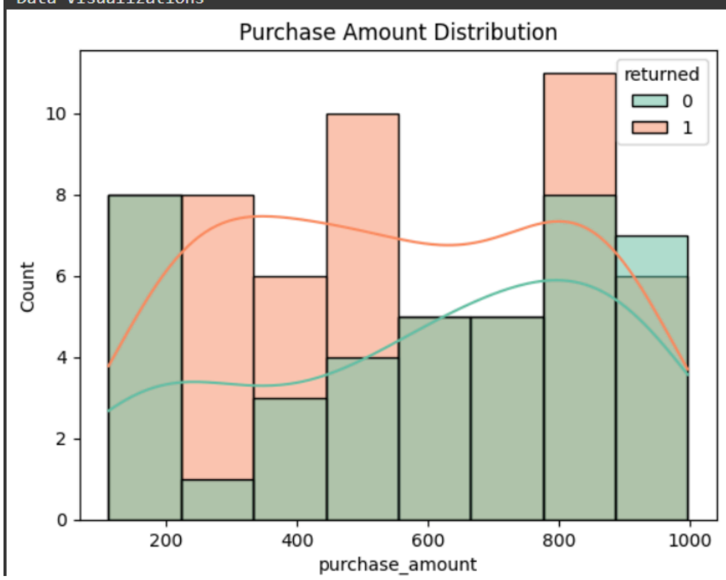
```
    print(" The product **will not be returned**.")

#Show visualizations
print("\n Data Visualizations")

#1. Purchase Amount Distribution
fig1, ax1 = plt.subplots()
sns.histplot(df, x='purchase_amount', hue='returned', kde=True, palette='Set2', ax=ax1)
plt.title("Purchase Amount Distribution")
plt.show()

#2. Review Score vs Return
fig2, ax2 = plt.subplots()
sns.boxplot(data=df, x='returned', y='review_score', palette='Set1', ax=ax2)
ax2.set_xticklabels(['No', 'Yes'])
plt.title("Review Score by Return Status")
plt.show()

#3. Delivery Days vs Return
fig3, ax3 = plt.subplots()
sns.violinplot(data=df, x='returned', y='days_to_delivery', palette='Set3', ax=ax3)
ax3.set_xticklabels(['No', 'Yes'])
plt.title("Delivery Days by Return Status")
plt.show()

#4. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
fig4, ax4 = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax4)
ax4.set_xlabel("Predicted")
ax4.set_ylabel("Actual")
```

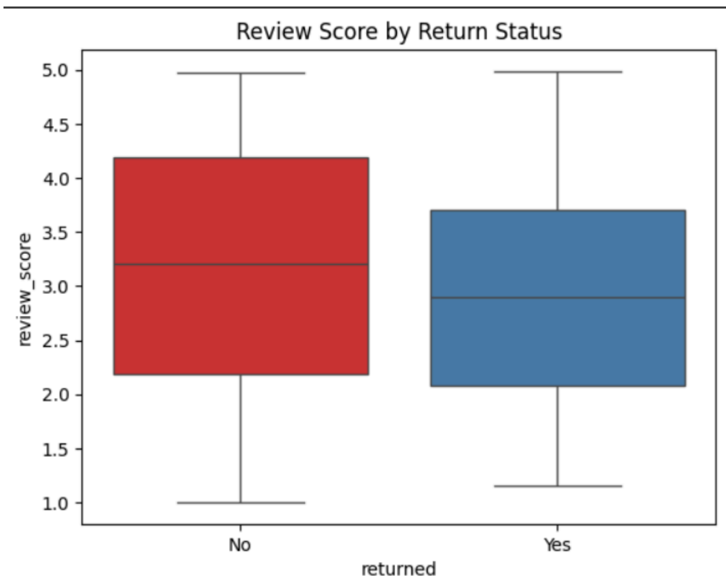
```
ax4.set_xticklabels(['No', 'Yes'])
ax4.set_yticklabels(['No', 'Yes'])
plt.title("Confusion Matrix")
plt.show()

#5. Classification report & Accuracy
print("\n Model Evaluation")
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

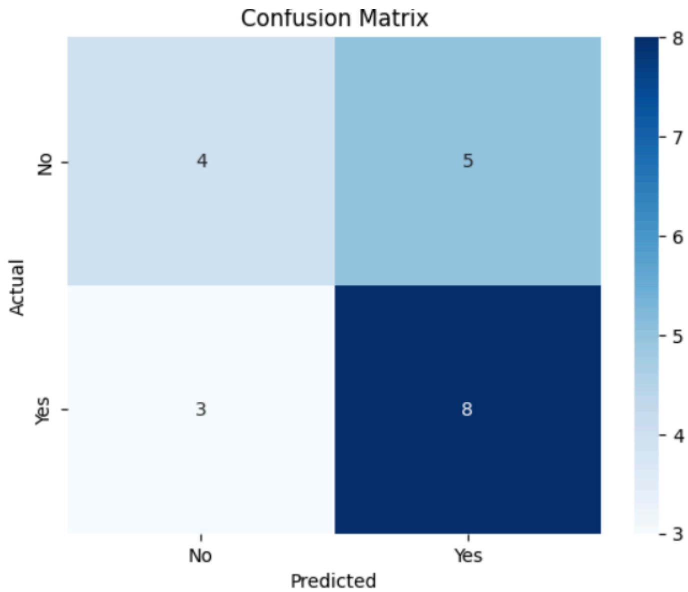
OUTPUT

```
 Product Return Prediction App
Enter Purchase Amount: 100000
Enter Review Score (0 to 5): 5
Enter Days to Delivery: 1
 The product **will not be returned**.
```

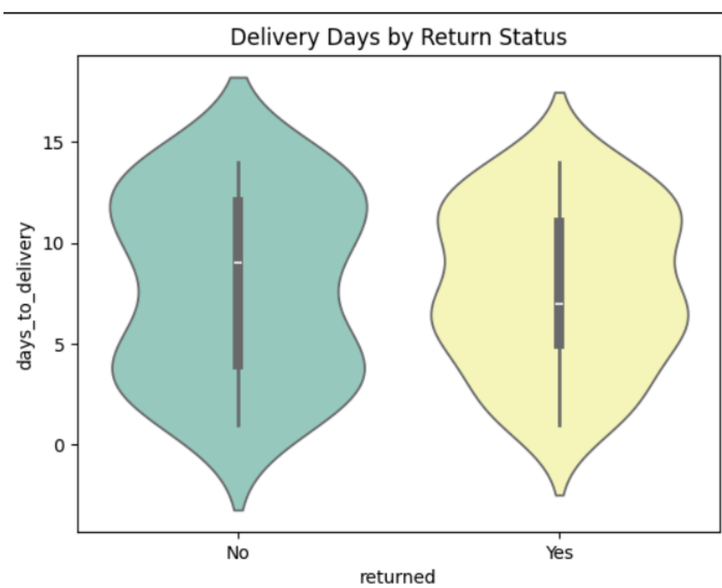
Data Visualizations








```
 Model Evaluation
Accuracy: 0.60
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.44      0.50         9
           1       0.62      0.73      0.67        11

    accuracy                           0.60        20
   macro avg       0.59      0.59      0.58        20
weighted avg       0.60      0.60      0.59        20
```



REFERENCES

- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
- *Product Return Dataset* (n.d.). Dataset used for model training and evaluation.
- Google. (n.d.). *Product Return Predictor*. GitHub repository. Retrieved from https://github.com/google/product_return_predictor
- AnFrBo. (n.d.). *Order Returns Prediction*. GitHub repository. Retrieved from https://github.com/AnFrBo/order_returns_prediction