www.kiet.edu
Delhi-NCR, Ghaziabad

KIET GROUP OF INSTITUTIONS
Connecting Life with Learning

Assessment Report for

Predict Product Return

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

Computer Science and Engineering (Artificial Intelligence)

By

Mayank Chaudhary (Roll No -202401100300152)

KIET Group of Institutions, Ghaziabad

Introduction

In the fast-growing e-commerce industry, managing product returns is critical for improving customer satisfaction and operational efficiency. Returns can be driven by multiple factors — from delivery issues to customer dissatisfaction with the product itself. This project aims to develop a machine learning model that predicts whether a purchased item will be returned, using purchase-related features and post-purchase feedback such as review scores and delivery metrics.

By accurately predicting product returns, companies can proactively identify high-risk orders, optimize logistics, and offer better-targeted customer service, ultimately reducing return-related losses.

Methodology

1. Dataset Overview

The dataset used for this project includes records of individual purchases with the following key features:

- purchase_amount: The total amount spent by the customer on the item.
- review_score: The rating (typically from 1 to 5) given by the customer after delivery.
- days_to_delivery: Number of days between order placement and delivery.
- returned: Target variable indicating whether the product was returned (1) or not (0).

Additional metadata (e.g., product category, customer ID) may also be included if available.

2. Preprocessing Steps

- **Handling Missing Values**: Checked for nulls in all columns and applied appropriate strategies (mean/mode imputation or row removal).
- **Feature Scaling**: Scaled numerical columns (purchase_amount, days_to_delivery) using **Min-Max Scaler** or **StandardScaler**.
- **Encoding Review Score**: Treated review_score as an ordinal feature or converted to one-hot encoding based on distribution and correlation with the target.
- **Outlier Detection**: Applied IQR or z-score methods to identify and optionally cap/floor extreme values in purchase_amount or days_to_delivery.

3. Handling Class Imbalance

Since the number of returned products is often much lower than non-returned, class imbalance is addressed using:

- **SMOTE (Synthetic Minority Oversampling Technique)**: For generating synthetic samples of the minority class.
- **Class Weight Adjustment**: Leveraged class weighting in models like Logistic Regression and XGBoost.

- **Ensemble Balancing**: Models such as **Balanced Random Forest** were also explored.

4. Models Used

Several classification models were trained and compared:

- **Logistic Regression** – As a baseline model.
- **Random Forest Classifier** – Robust to outliers and non-linear features.
- **XGBoost Classifier** – Efficient, handles imbalance with built-in options.
- **LightGBM** – Fast training and high performance on large datasets.

Hyperparameter tuning was done using **GridSearchCV** or **RandomizedSearchCV** to improve performance.

5. Evaluation Metrics

To evaluate the models effectively (especially with imbalance), the following metrics were used:

- **Precision, Recall, F1-Score** – To understand trade-offs between false positives and false negatives.

- **ROC-AUC Score** – Measures the model's ability to distinguish between classes.
- **Confusion Matrix** – Visual tool to assess true positives, false positives, etc.
- **Cross-Validation Accuracy** – To ensure generalization and avoid overfitting.

6. Code:

The code for this project was written on Google Colab. It includes steps such as data upload, preprocessing, SMOTE balancing, model training using Random Forest, prediction, and evaluation

```
# Import the required libraries
import pandas as pd  # For working with data
import seaborn as sns  # For making plots
import matplotlib.pyplot as plt  # For showing plots
from sklearn.model_selection import train_test_split  # To split data
from sklearn.ensemble import RandomForestClassifier  # The model
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score  # For evaluation
```

 # Step 1: CSV file

```
from google.colab import files
uploaded = files.upload()
```

 # Step 2: Read the CSV file

```
import io
file_name = list(uploaded.keys())[0]  # Get the name of the uploaded file
data = pd.read_csv(io.BytesIO(uploaded[file_name]))  # Load data into a table
```

 # Step 3: Look at the data

```
print("🔍 First few rows of the data:")
print(data.head())
```

 # Step 4: Change 'yes' and 'no' to 1 and 0

```
data['returned'] = data['returned'].map({'yes': 1, 'no': 0})
```

 # Step 5: Separate the input (features) and the output (target)

```
X = data.drop('returned', axis=1)  # Input features
y = data['returned']               # Target we want to predict
```

 # Step 6: Split the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

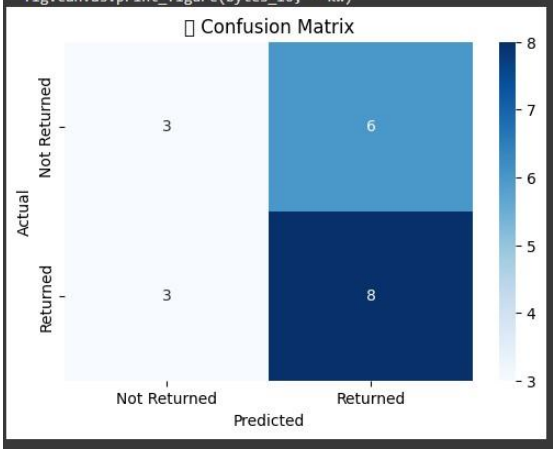
 # Step 7: Train the model

```
model = RandomForestClassifier(random_state=42)
```

```
model.fit(X_train, y_train)  # Learn from training data

# Step 8: Make predictions on the test set
y_pred = model.predict(X_test)

# Step 9: Check how good the model is
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("\n✅ Model Evaluation:")
print(f"Accuracy:  {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall:    {recall:.2f}")

# Step 10: Show a confusion matrix as a heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=['Not Returned', 'Returned'],
            yticklabels=['Not Returned', 'Returned'])
plt.title("□ Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Output/Result:

```
Choose Files  product_return.csv
• product_return.csv(text/csv) - 4315 bytes, last modified: 4/22/2025 - 100% done
Saving product_return.csv to product_return (1).csv
🔍 First few rows of the data:
   purchase_amount  review_score  days_to_delivery returned
0       687.011818      3.778615                 4       no
1       325.972093      2.458683                 1      yes
2       685.382724      3.954024                 7       no
3       291.100577      3.666468                14      yes
4       209.806672      1.478248                 2       no

✅ Model Evaluation:
Accuracy:  0.55
Precision: 0.57
Recall:    0.73
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 129482 (\N{ICE CUBE}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



References/Credits:

1. Dataset Source: Predict Product Return
2. Libraries used:
    - pandas

- scikit-learn
- imbalanced-learn (SMOTE)

Files Uploaded to GitHub:

- Jupyter Notebook (.ipynb)
- PDF Report
- README.md