Credit Card Default Prediction

Table of Contents

---

1. Introduction

This project involves building machine learning models for binary classification. The objective is to predict a target variable (`bad_flag`) based on various input features. The project utilizes a dataset containing customer transaction, onus, and bureau-related attributes.

---

2. Problem Statement

The objective is to develop a robust Behaviour Score for Bank A that predicts the likelihood of Credit Card customers defaulting. Using a development dataset with historical Credit Card details and default flags (bad_flag), the model will estimate default probabilities. Subsequently, this logic will be applied to a validation dataset to predict default probabilities for new accounts.

---

3. Data Description

Development Data:

A random sample of 96,806 Credit Card details.

Features Include:

1. **On-us Attributes**: Variables prefixed with "onus_attributes" (e.g., credit limit, repayment history).
2. **Transaction Attributes**: Variables prefixed with "transaction_attribute" (e.g., transaction counts, merchant-specific spending).
3. **Bureau Tradeline Attributes**: Variables prefixed with "bureau".

Additional Information:

1. **Target Variable (bad_flag)**: Indicates whether a customer has defaulted.
2. **Class Distribution**: Highly imbalanced, with a majority of non-defaulting accounts.

---

4. Exploratory Data Analysis (EDA)

EDA was performed to understand the dataset’s structure and patterns. Key findings include:

1. **Imbalance**: The dataset is highly skewed towards non-defaulting customers.
2. **Correlation**: Identified correlations between features and the target variable.
3. **Feature Insights**: Bureau attributes showed significant importance in predicting defaults.

Visualizations:

1. Distribution plots for transaction amounts and credit limits.
2. Heatmaps for feature correlation.
3. Count plots for class distribution.

---

5. Data Preprocessing

Steps:

1. **Data Cleaning**: Removed duplicates and handled missing values.
2. **Feature Scaling**: Standardized numerical attributes such as transaction counts and credit limits.
3. **Handling Categorical Variables**: Encoded categorical variables using one-hot encoding.
4. **Train-Test Split**: Split the dataset into 80% training and 20% testing data.

---

6. Handling Imbalanced Data

To address the severe class imbalance, Synthetic Minority Oversampling Technique (SMOTE) was used to oversample the minority class. SMOTE ensures the model learns patterns effectively without being biased towards the majority class.

Implementation:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)

X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
```

Result:

- Class distribution after SMOTE:
    - Non-defaulting: 48,403
    - Defaulting: 48,403

---

7. Model Building

Workflow:

1. **Data Exploration and Cleaning**:
    - Evaluated multiple columns, including customer account numbers, transactions, and bureau-related attributes.
2. **Visualization**:
    - Used libraries like Matplotlib and Seaborn to generate insights.
3. **Model Training and Evaluation**:
    - Trained multiple models, including Random Forest and XGBoost, for binary classification.

Models:

Random Forest Classifier

- Parameters:
    - Number of Trees: 100
    - Maximum Depth: None
- Results:
    - Accuracy: 79%
    - F1-Score: 0.73

XGBoost Classifier

- Parameters:
    - Learning Rate: 0.1
    - Number of Trees: 200
    - Maximum Depth: 6
    - Scale Pos Weight: Calculated based on class imbalance.
- Results:
    - Accuracy: 81%
    - F1-Score: 0.73

---

8. Evaluation Metrics

Given the class imbalance, the following metrics were used for evaluation:

- **Precision**: Fraction of predicted defaults that are correct.
- **Recall**: Fraction of actual defaults detected.
- **F1-Score**: Harmonic mean of precision and recall.
- **AUC-ROC**: Measures the tradeoff between true positive rate and false positive rate.

---

9. Conclusion

This project successfully developed a machine learning system to predict credit card defaults. By leveraging SMOTE for class imbalance and advanced algorithms like XGBoost, the system achieves high precision and recall, making it reliable for real-world deployment.

Key Takeaways:

- Addressing class imbalance is crucial for default prediction.
- XGBoost demonstrates superior performance for this type of dataset.

---

10. Future Scope

- **Real-Time Detection**: Deploying the model in a live system to flag potential defaults instantly.
- **Feature Engineering**: Incorporating domain-specific features to enhance model performance.
- **Explainability**: Adding SHAP or LIME for better interpretability of predictions.
- **Ensemble Methods**: Combining multiple models for improved robustness.
