Credit Default Prediction Model - User Manual

# 1. Introduction

## 1.1 Purpose

This guide explains the process of implementing and utilizing the Credit Default Prediction Model, which predicts the likelihood of a customer defaulting on credit card payments. The model assists financial institutions in assessing risk, improving lending decisions, and reducing defaults.

## 1.2 Scope

This repository contains the necessary code, documentation, and resources to execute the credit default prediction model. The guide covers the following areas:  
- Data Preprocessing  
- Exploratory Data Analysis (EDA)  
- Feature Engineering  
- Model Development using XGBoost  
- SHAP Value Interpretation

## 1.3 Key Terms and Abbreviations

- \*\*EDA\*\*: Exploratory Data Analysis  
- \*\*SHAP\*\*: SHapley Additive exPlanations  
- \*\*SMOTE\*\*: Synthetic Minority Over-sampling Technique  
- \*\*ROC-AUC\*\*: Receiver Operating Characteristic - Area Under Curve

# 2. Project Overview

## 2.1 Problem Statement

Credit default prediction is vital for financial institutions to minimize risks. Customers with lower credit limits and poor repayment histories are more likely to default. This model identifies at-risk customers early, enabling financial institutions to take appropriate actions.

## 2.2 Stakeholders

- Financial Institutions: Users who will apply the model for credit risk assessment and lending decisions.  
- Data Scientists: Developers responsible for model creation, data preprocessing, and feature engineering.  
- Business Analysts: Use model predictions to inform business strategies.  
- Customers: Individuals whose data is used for training and testing the model.

# 3. Requirements

## 3.1 Functional Requirements

The model should predict whether a customer will default on their credit card payments.  
The system should return predictions with a confidence score.  
The model should be easily integrated into existing financial systems.

## 3.2 Non-Functional Requirements

- \*\*Performance: Predictions should be made within 3 seconds.  
- Scalability: The model should handle large volumes of data.  
- Security: Sensitive customer data should be securely processed and stored.

# 4. Methodology

## 4.1 Data Collection and Preprocessing

### 4.1.1 Cleaning Data

The data is collected from a customer credit dataset and preprocessed through the following steps:  
- Missing Values: Imputed or removed based on the nature of the missing data.  
- Outliers: Detected and handled using IQR (Interquartile Range) or Z-scores.  
- Inconsistencies: Standardized categorical variables and transformed text data into useful features.

## 4.2 Exploratory Data Analysis (EDA)

- Credit Limit & Default Risk: Defaulters had significantly lower median credit limits than non-defaulters.  
- Repayment History: Repayment statuses strongly correlate with future payment behavior.  
- Age Distribution: Younger customers (20-30 years) showed a higher default rate.  
For detailed visualizations, check the Google Colab EDA notebook.

## 4.3 Feature Engineering for Default Prediction

Numerical features like credit limit, age, and repayment history were transformed and scaled. Feature importance was assessed using SHAP values.

## 4.4 Model Development

### 4.4.1 Implementing XGBoost

The XGBoost model was chosen due to its ability to handle structured data and class imbalance effectively.  
```python  
from xgboost import XGBClassifier  
model = XGBClassifier()  
model.fit(X\_train, y\_train)  
```

### 4.4.2 Performance Metrics

The model's performance is evaluated based on the following metrics:  
- Accuracy  
- Precision  
- Recall  
- F1-Score  
- ROC-AUC

## 4.5 SHAP Value Integration & Interpretation

### 4.5.1 Explaining Model Predictions Using SHAP

SHAP was integrated to explain model predictions by identifying the contribution of each feature.

### 4.5.2 New Features Created After SHAP Analysis

Features such as repayment behavior over time were derived based on SHAP insights to improve interpretability.

### 4.5.3 Visualizing Feature Impact

SHAP Waterfall and Impact plots were used to visualize feature contributions and their impact on model predictions.

### 4.5.4 Addressing Class Imbalance

SMOTE (Synthetic Minority Over-sampling Technique) was employed to generate synthetic data for the minority class.

## 4.6 Model Fairness Testing

### 4.6.1 Assessing Model Fairness

Fairness testing was conducted using metrics like equal opportunity and demographic parity to ensure the model's fairness across different groups (e.g., age, gender).

### 4.6.2 Libraries and Techniques

Fairness testing utilized libraries like AIF360 and Fairness Indicators.

# 5. Model Evaluation

## 5.1 Model Validation and Cross-Validation

Cross-validation was used to validate the model, ensuring its ability to generalize across different subsets of data.

## 5.2 SHAP Insights and Adjustments

SHAP insights helped refine the model by adjusting it to focus on high-impact features and removing unnecessary complexity.

# 6. Conclusion and Recommendations

## 6.1 Conclusion

The model successfully predicts credit default based on features like credit limits and repayment history. SHAP adds transparency, making model decisions understandable for stakeholders.

## 6.2 Recommendations

- Continuous Updates: Retrain the model with new data to maintain its accuracy.  
- Feature Expansion: Incorporate additional features such as spending behavior to enhance predictions.  
- Real-Time Predictions: Implement real-time predictions for dynamic risk assessment.

# 7. Future Scope

## 7.1 Future Scope

- Deep Learning Integration: Consider deep learning models for improved prediction accuracy.  
- Alternate Data Sources: Explore integrating social media or other external data sources for enhanced predictions.  
- Production Deployment: Scale the model for production environments and improve deployment pipelines for real-time capabilities.

# 8. Installation Guide

## 8.1 Prerequisites

- Python 3.x  
- Required libraries (listed in `requirements.txt`)

## 8.2 Installation Steps

1. Clone this repository:  
  
 ```bash  
 git clone https://github.com/ryancodingg/predicting-default-risk-using-shap/blob/main/Project2\_TAP.ipynb

```  
  
2. Install the dependencies:  
  
 ```bash  
 pip install -r requirements.txt  
 ```  
  
3. Run the model:  
  
 ```bash  
 python run\_model.py  
 ```  
  
Alternatively, visit the Google Colab notebook - https://colab.research.google.com/github/ryancodingg/predicting-default-risk-using-shap/blob/main/Project2\_TAP.ipynb

# 9. License

This project is licensed under the MIT License. See the `LICENSE` file for more details.

# 10. Acknowledgments

- Neubauer, D., et al. (2022). \*Understanding Credit Risk Prediction Models: Insights and Applications\*. Journal of Financial Technology, 45(2), 112-130.  
- SHAP documentation: [SHAP Docs](https://shap.readthedocs.io)  
- SMOTE implementation: [imbalanced-learn](https://imbalanced-learn.org)

- Google Collab - https://colab.research.google.com/github/ryancodingg/predicting-default-risk-using-shap/blob/main/Project2\_TAP.ipynb