LOAN DEFAULT RISK CLASSIFICATION FOR A MICROFINANCE STARTUP USING PYTHON-BASED MACHINE LEARNING

1. Introduction

A fintech startup in the microfinance sector aimed to automate its loan application screening process. The team faced challenges in identifying potentially high-risk borrowers early, leading to a growing default rate and manual review delays. They needed a Python-based machine learning solution that could classify loan applications into “likely to default” and “likely to repay” categories based on historical borrower data.

The classification model would assist loan officers in prioritizing applications and allocating manual review efforts more efficiently, while improving risk-adjusted decision-making across loan products.

2. Objective

- To create a supervised classification model using Python to flag potential defaulters before loan approval
- To evaluate and compare multiple algorithms and select the most robust model
- To provide the client with both technical code and a clear visual report for executive understanding and operational use
- To suggest business rules derived from the model for integration into their loan approval logic

3. Data Used

The client shared a dataset of **7,500 loan applications** with loan outcomes (default or not) for the last 18 months. Fields included:

- Loan_Status – Target variable (1 = Repaid, 0 = Defaulted)
- Applicant_Income
- Loan_Amount
- Tenure (in months)
- Credit_Score

- Employment_Status (Salaried / Self-Employed)
- Residence_Type (Owned / Rented)
- Previous_Default (Yes / No)
- Marital_Status (Married / Single)

Dummy variables were created for categorical fields, and the target class was slightly imbalanced (~78% repaid, 22% defaulted).

4. Methodology

4.1 Data Preprocessing

- Converted categorical variables using one-hot encoding
- Used SMOTE to balance classes by generating synthetic minority class samples
- Performed standard scaling for continuous variables
- Created new feature: **Debt-to-Income Ratio**
- Identified and removed duplicate records and entries with incomplete credit history

4.2 Model Development and Evaluation

Compared multiple classification models:

- Logistic Regression
- Random Forest
- XGBoost
- Support Vector Machine

Used an 80/20 train-test split with **stratified sampling** and applied:

- Cross-validation (5-fold)
- Evaluation Metrics:
    - Accuracy
    - Precision
    - Recall
    - F1-Score
    - ROC-AUC Score

4.3 Tools Used

- Python IDE: Jupyter Notebook
- Libraries: pandas, numpy, scikit-learn, xgboost, matplotlib, seaborn, imblearn
- Delivered code ready for re-use in **Google Colab and VS Code**

5. Key Results

- **Best Model**: Random Forest Classifier
- **Accuracy**: 86.4%
- **Precision** (on Default class): 85.1%
- **Recall** (on Default class): 81.7%
- **F1-Score** (on Default class): 83.4%
- **AUC-ROC**: 0.91

Top Predictors (based on feature importance):

- Credit Score
- Debt-to-Income Ratio
- Previous_Default
- Loan Amount
- Employment_Status (self-employed applicants had higher risk)

6. Delivered Report Summary

- Python Notebook with all code, model comparisons, and re-training pipeline
- CSV with predictions on test set and probability scores
- PDF Report (12 pages) including:
    - Model architecture diagram
    - Confusion matrices and ROC curves
    - Explanation of evaluation metrics and what they mean for business
    - Decision logic that can be converted into business rules
- PPT Slide Deck for internal presentation to board/investors

7. Business Impact

- Enabled the client to **automatically flag ~90% of high-risk applicants** before manual underwriting
- Reduced manual review load by over 50%, allowing credit officers to focus on marginal cases
- Helped the company reduce new default cases by **17%** within the first quarter of deployment
- Offered data-backed justifications to deny or defer applications, improving transparency with applicants

8. Future Scope

- Integrate the model with the loan origination system (LOS) using a lightweight Flask API
- Include additional external data such as mobile phone usage or utility payment history
- Implement a feedback loop to retrain the model quarterly using new outcomes
- Transition to a **multi-class risk scoring model** instead of binary classification
- Expand the model to cover cross-sell opportunities based on repayment behavior