Microcredit Payback Prediction – Project Report

# Introduction

This project builds a probability-based classifier to predict whether a customer will repay a microcredit within 5 days of issuance.

The use case sits at the intersection of telecom and inclusive finance: enabling instant, low-ticket credit to value-conscious, low-income subscribers while managing portfolio risk.

Data science helps by transforming raw transaction and subscriber data into decisions — prioritizing the right customers, setting credit policies, and reducing losses while expanding access.

# Problem Statement

A telecom operator collaborating with a Microfinance Institution (MFI) offers micro-loans on mobile balances with repayment due in 5 days.

Each loan transaction is labeled: 1 = paid back (non-defaulter), 0 = not paid (defaulter).

Objective: predict the probability of repayment for each transaction to improve customer selection and credit operations.

Success entails accurate, calibrated probabilities that perform well on Log Loss, while keeping strong Recall (catching likely defaulters) and solid Precision (avoiding over-rejection).

# Data Collection and Preprocessing

Dataset: 'Micro-credit-Data-file.csv' (loaded in the notebook). Columns include numeric and categorical features describing the subscriber and transaction.

Preprocessing steps implemented in the notebook:

• Missing values: numeric imputed with mean, categorical imputed with most frequent (SimpleImputer).

• Categorical encoding: LabelEncoding for binary categories; One-Hot Encoding (pd.get\_dummies) for multi-class categories.

• Train/Validation/Test: Stratified splits (train\_test\_split, StratifiedKFold) to preserve class balance.

• Scaling: StandardScaler applied to numerical features for distance- and margin-based models (e.g., Logistic Regression, SVM, kNN, SGD).

# Exploratory Data Analysis (EDA)

The notebook inspects missingness and class balance and draws basic univariate/bivariate plots (seaborn/matplotlib).

EDA focuses on relationships between user/transaction attributes and outcomes (repayment within 5 days).

Examples typically include: repayment rate by top categorical levels, numeric distributions split by target, and correlation heatmaps for numeric blocks.

# Feature Engineering

The notebook creates engineered variables (Cell 4 and Cell 10), including interaction-style or ratio-style features derived from raw columns, and consolidated encodings.

Examples of useful constructs in this domain often include:

• Recent behavior aggregates (counts/means over windows)

• Usage intensity ratios (e.g., top-ups per day, ARPU-like proxies)

• Recency features (days since last top-up/loan)

• Loan-amount related transforms (nonlinearities or bins)

These features aim to improve separability of payers vs non-payers and provide lift to tree-boosting models.

# Model Selection and Training

The notebook trains a wide model zoo (≈45 variants) spanning:

• Linear/Margin: LogisticRegression, SGDClassifier, RidgeClassifier, PassiveAggressive, Perceptron

• Tree/Bagging: DecisionTree, RandomForest, ExtraTrees, BaggingClassifier

• Boosting: GradientBoosting, HistGradientBoosting, AdaBoost, XGBoost, LightGBM, CatBoost

• Distance/Probabilistic: KNeighbors, GaussianNB, BernoulliNB, QDA/LDA

• Kernels: SVC (probability=True)

Training uses stratified CV. A Column-aware preprocessing (imputation/encoding/scaling) is applied before model fitting.

For probabilistic evaluation, predict\_proba is used where supported; otherwise calibrated approaches are considered.

# Hyperparameter Tuning

Both GridSearchCV and RandomizedSearchCV are used across families with scoring='neg\_log\_loss' and stratified K-folds.

Illustrative grids include:

• LogisticRegression: C, penalty, class\_weight

• Tree/Forest: max\_depth, n\_estimators, min\_samples\_split, max\_features

• Boosting (XGB/LGBM/CatBoost/HistGB): learning\_rate, depth/max\_depth, n\_estimators, subsample/colsample

• SVC: C, gamma, kernel; probability=True

The best estimators per family are retained for final comparison.

# Model Evaluation

Primary metrics computed on the held-out test set (see final comparison table in the notebook):

• Log Loss (lower is better; measures probability calibration and confidence)

• Precision (positive class = repaid) – share of predicted repayers that truly repay

• Recall (positive class = repaid) – share of actual repayers correctly identified

The code prints a sorted comparison by LogLoss and also the classification\_report for selected finalists.

(Exact numbers depend on the specific dataset instance; run the notebook with 'Micro-credit-Data-file.csv' present to reproduce.)

# Feature Importance Analysis

For interpretability, the notebook extracts importances from tree-based models and gradient boosted trees and displays the top features:

• RandomForest: feature\_importances\_

• XGBoost: feature\_importances\_ / gain-based importance

• HistGradientBoosting: feature\_importances\_ (where available)

These help identify which subscriber and transaction signals most strongly drive repayment probability (e.g., recent behavior aggregates, amount-related features, and usage recency variables).

# Business Implications

• Smarter Eligibility & Limits: Use predicted probabilities to approve, decline, or set credit limits; tighten policy on low-probability segments.

• Dynamic Pricing: Align fees/interest with risk to keep portfolio-level economics sustainable.

• Collections Prioritization: Focus reminders and nudges on loans with moderate probability (most influenceable) to improve payback.

• Growth with Guardrails: Expand access while constraining expected loss through probability thresholds and portfolio monitoring.

• A/B Test Policies: Continuously test cutoffs and offer sizes to maximize expected margin and customer satisfaction.

# Conclusion and Future Steps

Outcomes: A robust, probability-calibrated classifier with strong LogLoss and balanced Precision/Recall, plus interpretable feature importance for policy-making.

Limitations: Availability of only tabular, transaction-level data; potential dataset shift; cost-sensitive thresholds not yet optimized to business KPIs.

Next steps:

• Add calibration (Platt/Isotonic) and expected profit curves to pick operating thresholds.

• Incorporate temporal cross-validation and richer recency/trajectory features.

• Deploy as an API; monitor drift; retrain on rolling windows.

• Expand ensembling (stacking) and try monotonic constraints for stability on boosting models.