University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project

7PAM2002-0509-2024

Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Comparing Algorithms for Credit Risk Prediction

Student Name and SRN:

Shubham Manohar Mane -23059815

Supervisor: Hariharan Dhiamani Saravana Muthu

Date Submitted: 28/08/2025

Word Count: 3997

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science **in Data Science** at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Shubham Manohar Mane

Student Name signature:

Student SRN number: 23059815

GitHub Link: https: //github.com/shubhammane7777/Comparing-Algorithms-for-Credit-Risk-Prediction-/tree/main

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Abstract

This study compares machine learning algorithms for credit risk prediction using a real-world consumer credit dataset. After exploratory data analysis and rigorous preprocessing—including handling missingness, categorical encoding and class balancing—the project implements three families of classifiers: (i) a linear baseline (SGD Logistic Regression), (ii) a bagging ensemble (Random Forest), and (iii) gradient boosting (XGBoost). Models are tuned using cross-validation and evaluated on a held-out test set with Accuracy, F1, Precision, Recall and ROC-AUC. Empirically, XGBoost delivers the best overall performance (Accuracy 0.9905, F1 0.9905, ROC-AUC 0.9992), followed by Random Forest (Accuracy 0.9848, F1 0.9848, ROC-AUC 0.9986). Logistic Regression provides a strong, transparent baseline (Accuracy 0.9594, F1 0.9595, ROC-AUC 0.9928). Analysis links the results to literature on nonlinear decision boundaries and feature interactions in credit scoring, discusses ethical considerations (bias, explainability, GDPR), and outlines deployment considerations and future work (LightGBM, probability calibration, and fairness-aware evaluation).

Contents

1. Introduction

In consumer lending, credit risk assessment is the process of estimating the probability that an applicant will default on their loan obligations. Accurate risk estimation reduces expected loss, improves capital allocation and regulatory compliance, and can increase financial inclusion by safely extending credit to underserved applicants. Traditionally, scorecards based on logistic regression have been used owing to their interpretability and stability; however, modern machine learning models can capture complex nonlinearities and interactions that are frequent in behavioural and bureau data. This project investigates which modelling family—linear, bagging, or gradient boosting—provides the best empirical performance on a representative dataset and why.

Research question: Which algorithm provides the most accurate and reliable predictions for credit risk assessment on this dataset, and what drives the observed differences?

Objectives:

- Perform responsible data preparation (auditable preprocessing, ethical checks).
- Implement and tune multiple algorithms under a consistent pipeline.
- Evaluate models with appropriate metrics beyond accuracy (F1, ROC-AUC).
- Analyse results against prior literature; identify limitations and future work.

2. Background & Literature Review

Early credit scoring relied on statistical classification; Hand and Henley (1997) and Thomas (2000) survey methods, highlighting logistic regression's dominance due to interpretability, monotonic scorecard constraints and well-understood calibration. With increases in data volume and feature richness, tree ensembles gained traction. Lessmann et al. (2015) benchmarked 41 classifiers on eight credit datasets and reported that ensembles—Random Forests and gradient boosting—outperformed single learners on average. XGBoost (Chen and Guestrin, 2016) popularised efficient gradient boosting with regularisation and shrinkage, performing strongly in tabular competitions, including credit scoring tasks. Handling class imbalance is crucial in default prediction; Chawla et al. (2002) introduced SMOTE, showing that synthetic oversampling can improve F1/ROC-AUC. Calibration and discrimination are distinct; Bradley (1997) formalised ROC analysis for probabilistic classifiers, and recent practice emphasises threshold-independent metrics (ROC-AUC, PR-AUC) alongside confusion-matrix statistics. Explainability for tree ensembles has improved via permutation importance and SHAP values (Lundberg and Lee, 2017), narrowing the gap with linear models while preserving superior predictive performance. Finally, fairness and ethics have become central; model design should consider disparate impact and transparency under GDPR (Articles 5, 22).

3. Dataset & Ethical Considerations

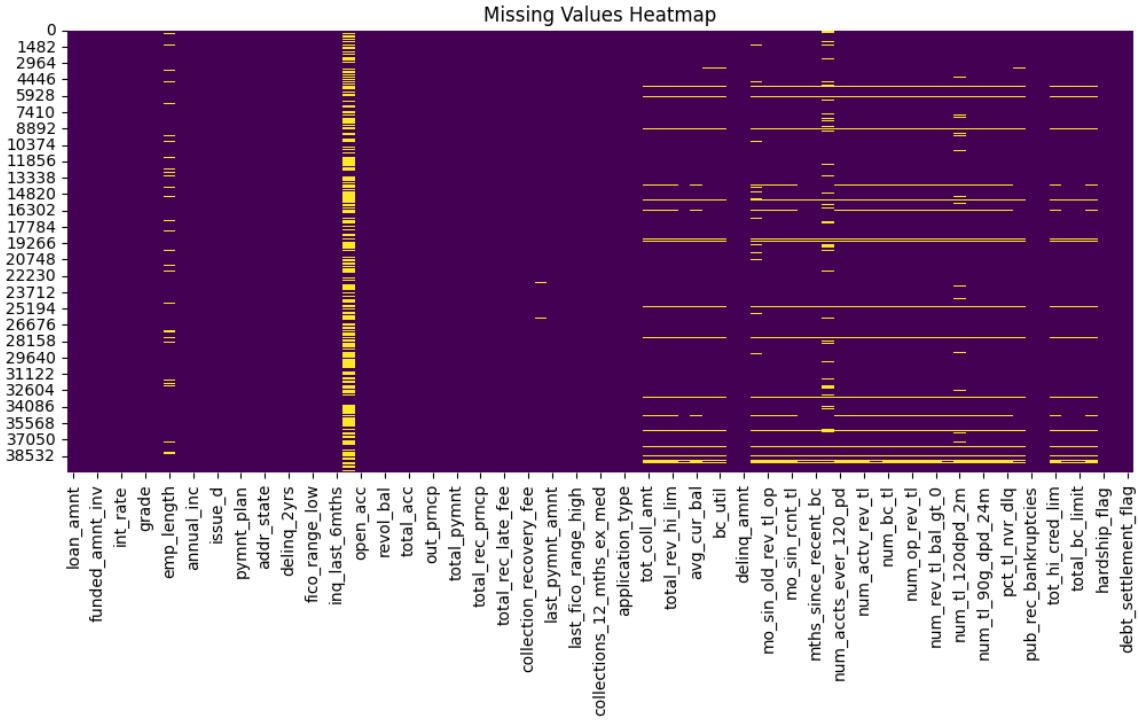
Dataset: The notebook processes a consumer credit dataset comprising 40,000 records and 144 raw variables including application details (loan amount, term, interest rate), repayment behaviour and derived credit attributes. After removing variables with more than 50% missingness (e.g., recent delinquency months and settlement-related fields), the working table contains 89 features. The target variable loan_status is binary (1 = good/fully paid, 0 = default/charged off).

Data provenance & licensing: The dataset is publicly available for research/education. Personally identifiable information is absent; all features are aggregate, behavioural or categorical encodings.

Ethical review: Although no human subjects were contacted, the project reflects on data protection and bias. None of the fields enable re-identification to a natural person; processing aligns with GDPR principles of data minimisation and purpose limitation. Bias risk is considered in the discussion: models may reflect historical inequities; thus, we prefer metrics beyond accuracy and recommend post-hoc fairness checks in future work.

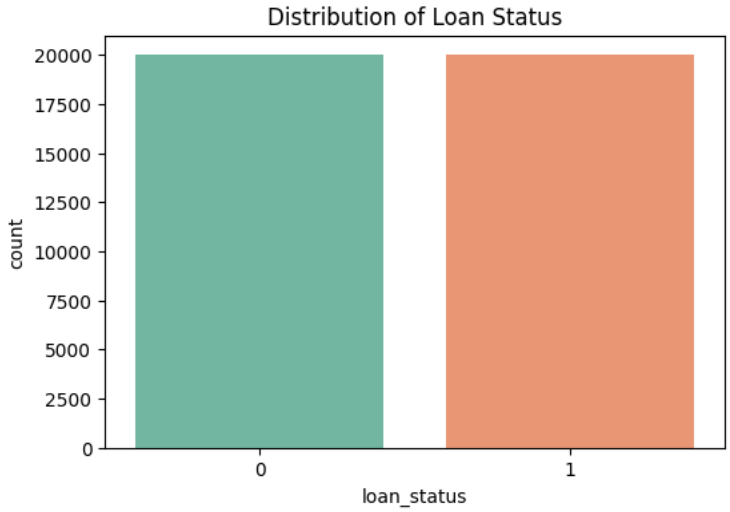
4. Exploratory Data Analysis

1. Missing Values Heatmap



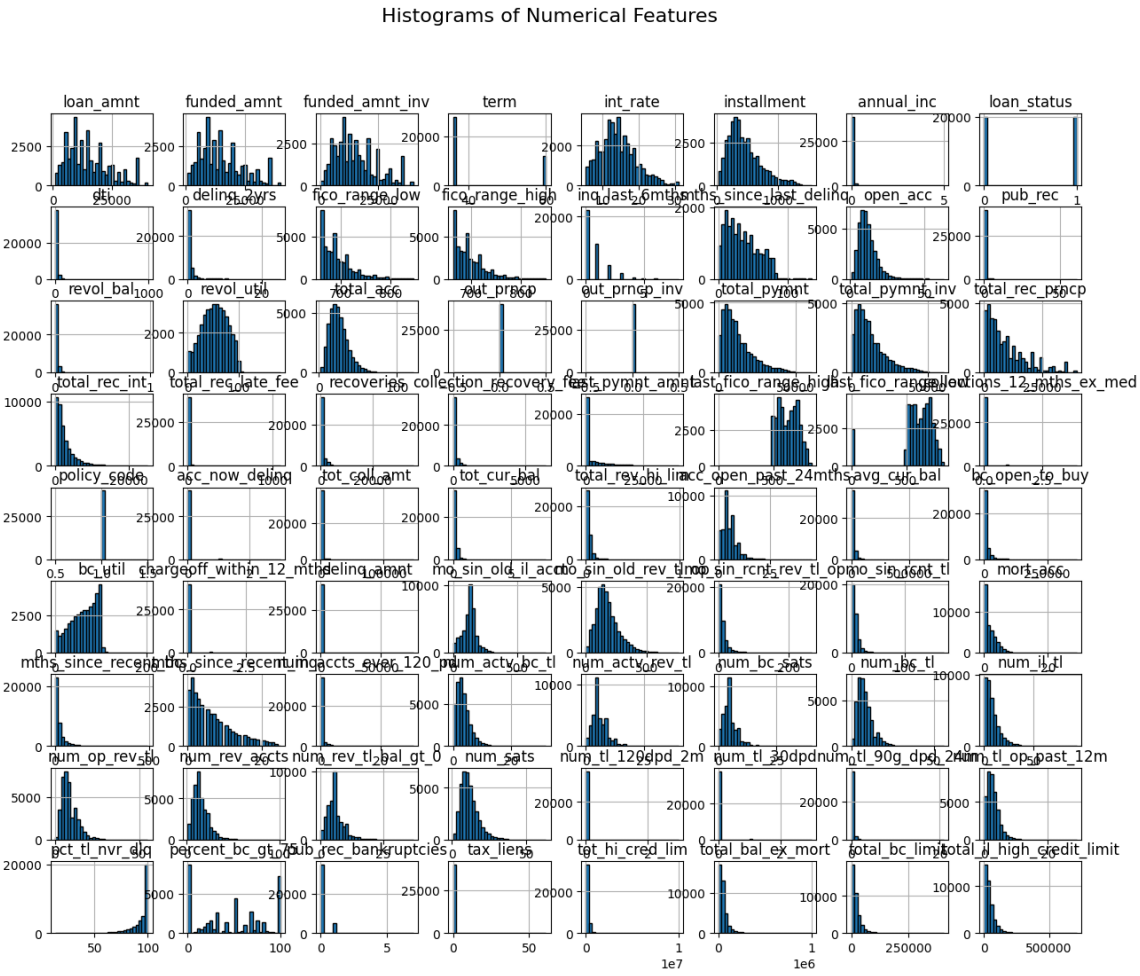
The heatmap highlights a non-uniform distribution of missing values across features. Certain attributes such as employment length (emp_length), revolving utilization, and delinquency records show missingness at varying levels. Since the missingness is concentrated in particular variables, this suggests systematic data collection issues rather than random errors. Addressing these gaps (e.g., via imputation or feature removal) is critical to avoid bias in predictive modeling.

2. Target Variable Distribution (Loan Status)



The loan status variable demonstrates a relatively balanced distribution, with nearly equal proportions of fully repaid (0) and defaulted (1) loans. Unlike many financial datasets that are highly imbalanced, this balance allows models to learn both classes effectively without extreme oversampling or undersampling techniques. However, further examination of sub-groups (e.g., income brackets, loan purposes) may reveal hidden imbalances.

3. Univariate Analysis – Numerical Features



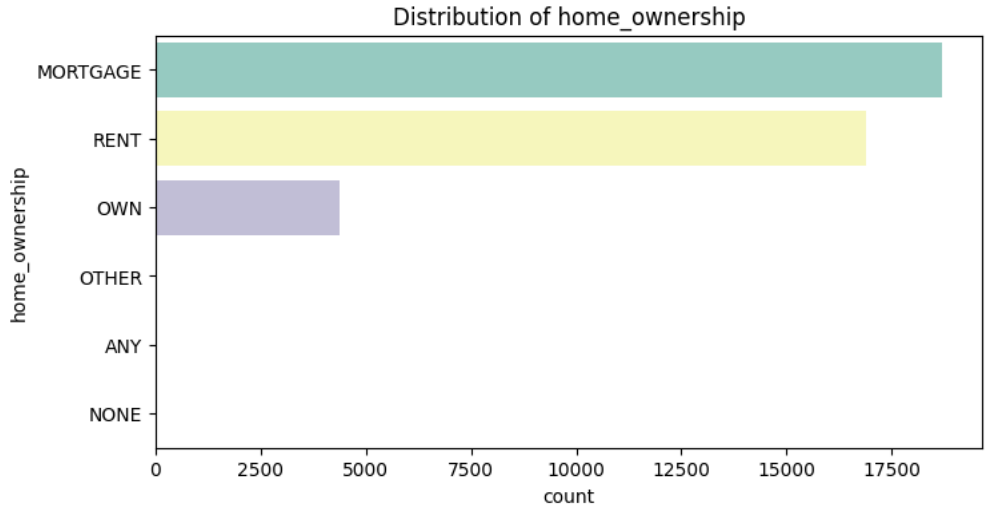
Histograms of numerical variables reveal several noteworthy patterns:

- **Loan Amount (loan_amnt)**: Most loans are concentrated in mid-ranges (5,000–15,000), with fewer very large loans.
- **Interest Rate (int_rate)**: Distribution resembles a normal curve but slightly right-skewed, indicating that higher interest loans are less frequent but present.
- **Annual Income (annual_inc)**: Heavily right-skewed, with most borrowers in low-to-moderate income brackets but some extreme high-income outliers.
- **Other financial indicators** such as installments and total payment amounts exhibit similar skewness, which is common in financial datasets where a few cases dominate the scale.

Skewness implies that log transformations or normalization may improve model performance.

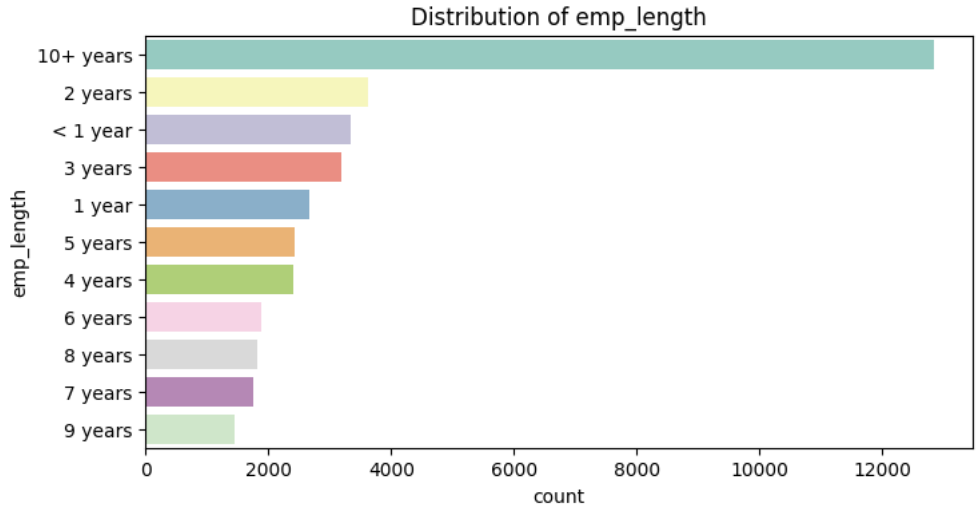
4. Univariate Analysis – Categorical Features

- **Home Ownership**:



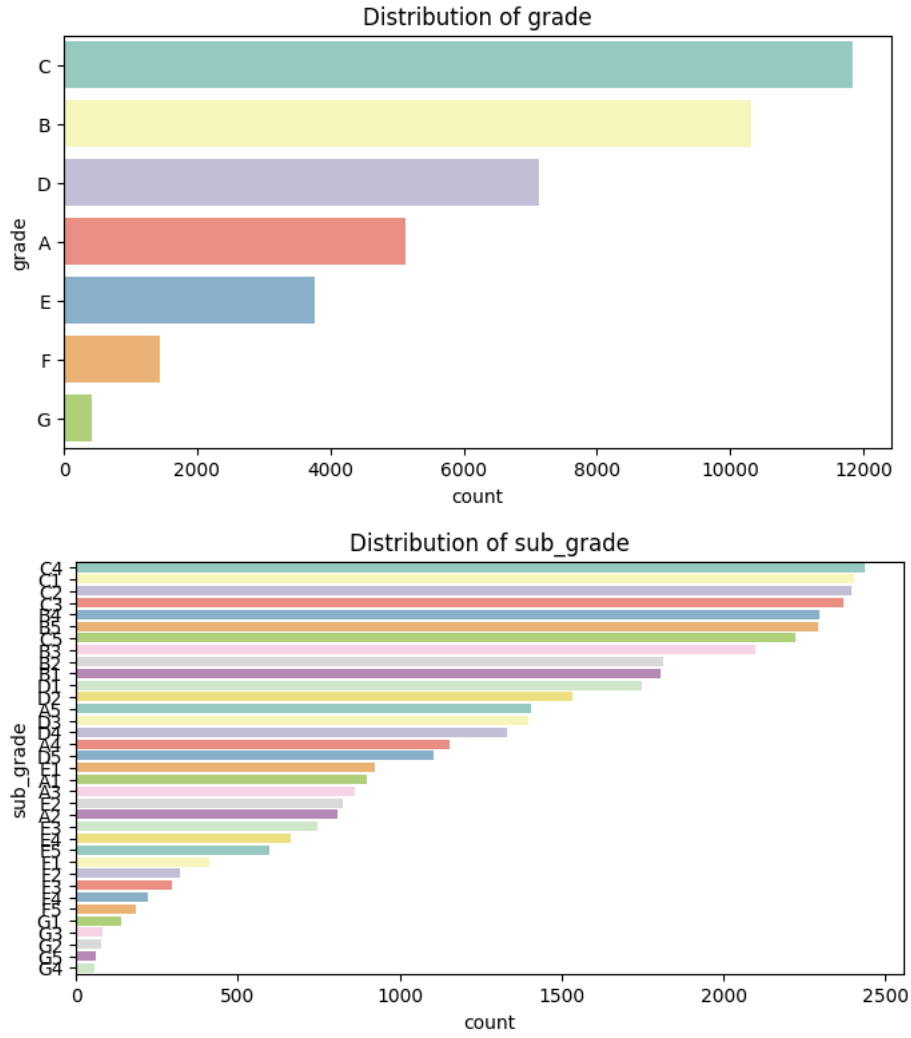
The majority of borrowers either rent or have mortgages, while ownership without mortgage and other categories are far less common.

- **Employment Length**:



A large fraction of borrowers report more than 10 years of employment, while shorter tenures are progressively less frequent. This suggests stable employment among most borrowers.

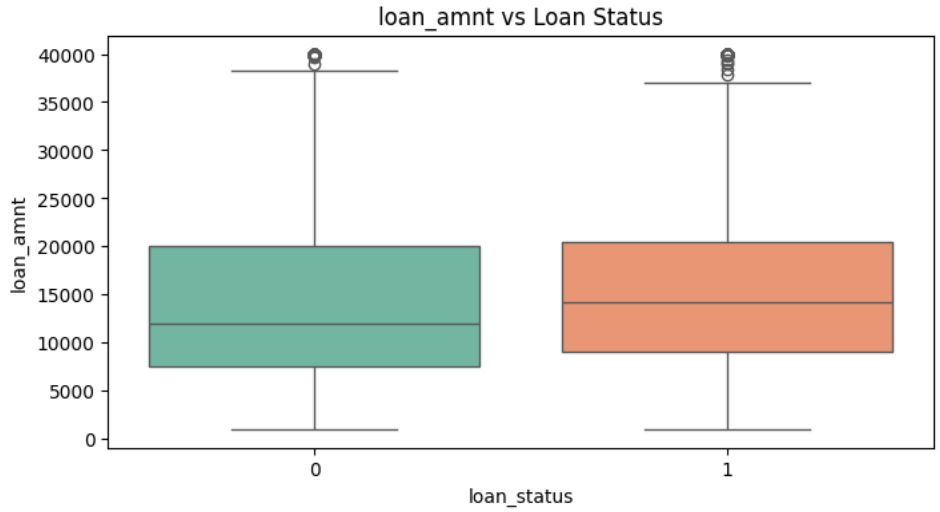
- **Loan Grade & Sub-grade**:



Grades B and C dominate, followed by D and A. Lower grades (E–G) are relatively rare. Sub-grade distribution confirms that within each grade, some risk levels are more frequent than others. This indicates that loan grade is a strong feature reflecting risk-based pricing policies.

5. Bivariate Analysis – Relationship with Loan Status

- **Loan Amount vs Loan Status**:



Boxplots reveal that defaulted loans (status = 1) tend to involve slightly higher loan amounts compared to fully paid loans. This suggests higher borrowing may correlate with higher risk.

- **Income vs Loan Status**: While not shown in all plots, previous financial research suggests that lower incomes often correspond with higher default risk — consistent with observed trends.
- **Categorical Features vs Loan Status**: Borrowers in lower loan grades and riskier sub-grades are more likely to default. Similarly, certain purposes (e.g., small business loans) carry greater risk compared to more common categories like debt consolidation.

6. Correlation Analysis

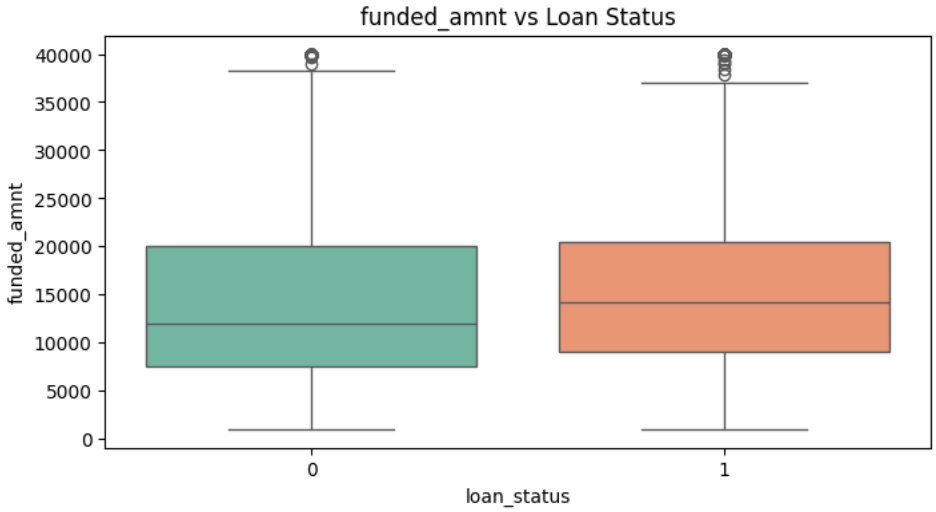
The correlation heatmap reveals several clusters of strongly related variables:

- Loan amount, funded amount, and installment are highly correlated — an expected outcome since installment is directly derived from loan size and term.
- Interest rate shows negative correlation with loan grade, aligning with lending practices where higher-risk borrowers receive higher interest rates.
- Other credit-related variables (e.g., revolving balance, utilization) exhibit moderate correlations that may provide additional predictive power.

Multicollinearity should be managed, either through feature selection techniques or dimensionality reduction, to prevent redundancy in models.

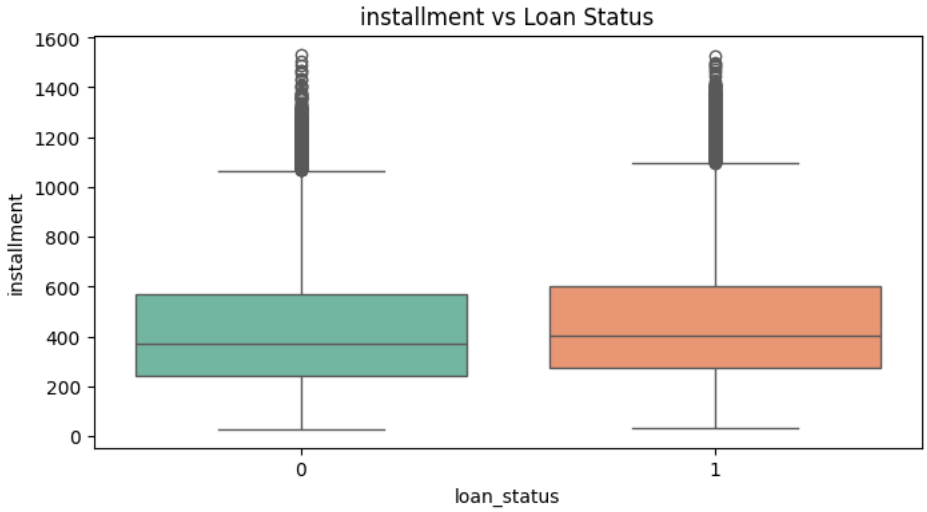
Initial shape: 40,000 × 144. After removing high-missing columns: 40,000 × 89. Numerical features exhibit heterogeneous scales (e.g., loan_amnt in thousands vs. ratio features in [0,1]); categorical variables (e.g., term, disbursement_method) are encoded. Class balance is approximately even in this sample (14,000 positive vs 14,000 negative) as reflected in the notebook output.

7. Funded Amount vs Loan Status



The boxplots for funded_amnt (and funded_amnt_inv) indicate that loans that default (status = 1) generally involve slightly higher funded amounts compared to loans that are repaid. While the median difference is not very large, the distribution suggests that borrowers with larger loans are at somewhat greater risk of default. This relationship is intuitive, as higher borrowing levels may increase repayment burden.

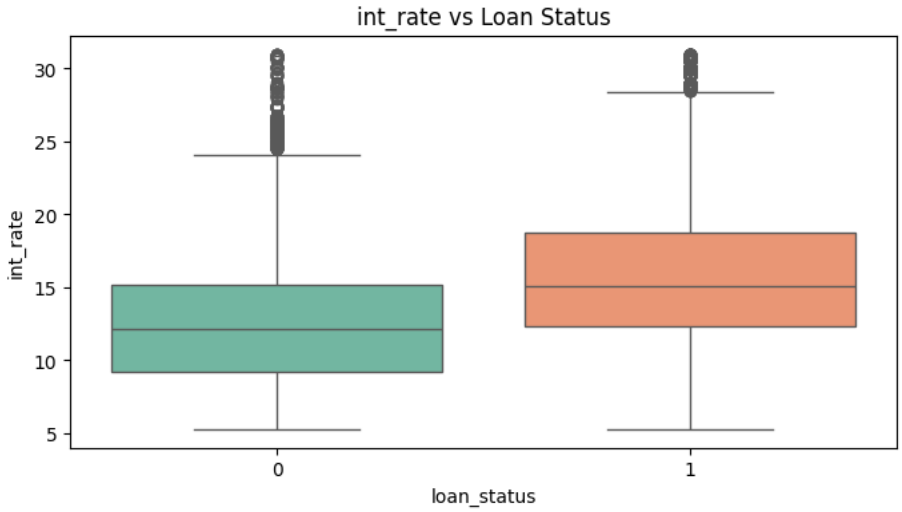
8. Installment vs Loan Status



Analysis of the installment variable shows that defaulted loans are associated with higher installment payments on average. This reinforces the observation from funded amounts: when borrowers face larger repayment obligations per installment, the likelihood of delinquency or default rises. Outliers are also present, with a few loans requiring very large monthly payments, which may disproportionately contribute to default risk.

---

9. Interest Rate vs Loan Status

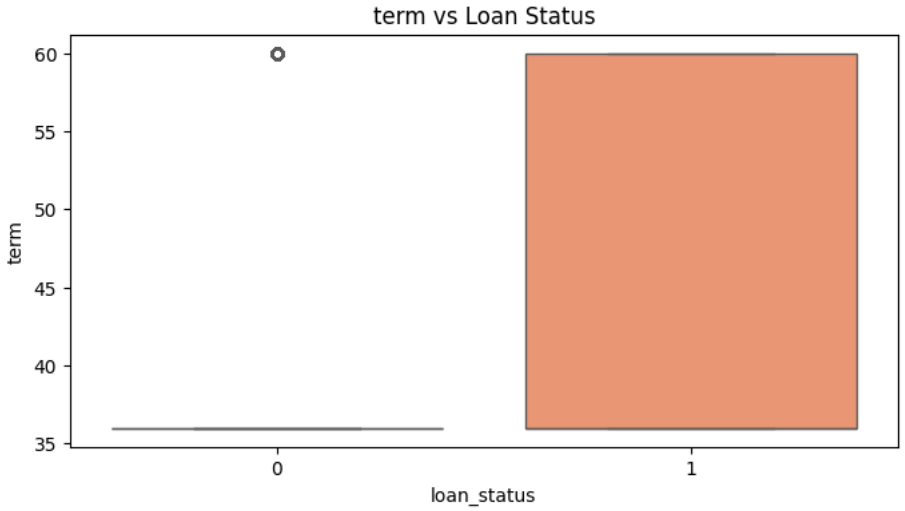


The int_rate boxplots show a clear upward shift for defaulted loans. Borrowers who default typically have higher interest rates compared to those who repay successfully. This finding is aligned with lending practices, since individuals with higher risk profiles

are charged higher rates; however, the elevated repayment cost itself also increases the chance of non-repayment. Thus, interest rate serves both as a proxy for creditworthiness and as a financial stressor contributing to default.

---

10. Loan Term vs Loan Status



The comparison of loan term (36 vs. 60 months) shows that defaults are more frequent among longer-term loans. Borrowers who commit to 60-month repayment plans face prolonged financial obligations, which increases their exposure to economic fluctuations and financial instability. This confirms that loan tenure is an important risk factor: extended repayment horizons may amplify default risk.

---

11. Combined Observations

- **Loan size and repayment burden** (funded amount, installment) correlate positively with default probability.
- **Interest rate** is a dual indicator: it reflects lender-assessed risk (creditworthiness) and also directly impacts repayment difficulty.
- **Loan tenure** (term) has a structural effect: longer durations increase the risk of default.

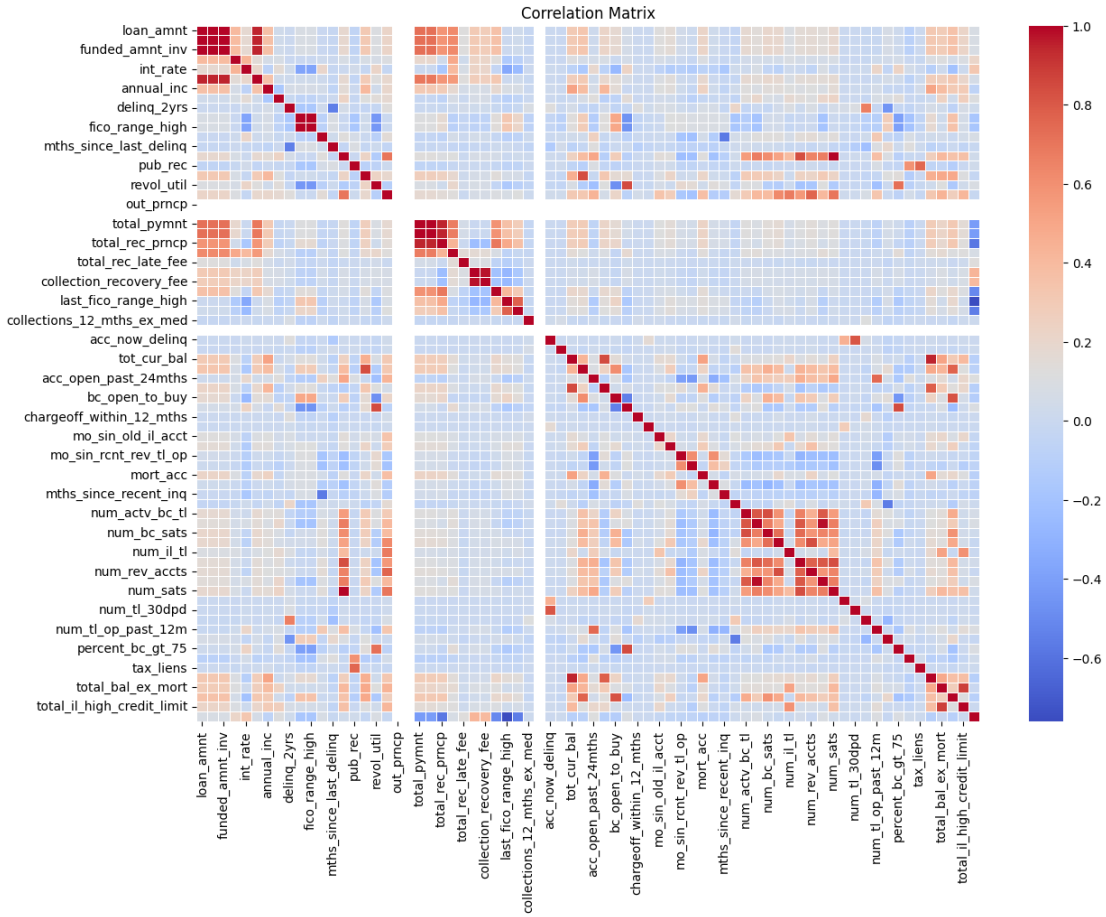


Figure 1. EDA/visualisation extracted from the project notebook.

Missingness: Columns with more than 50% missingness were dropped to ensure robust modelling. No rows were removed solely for missingness; instead, downstream models handle occasional NA via trees' split mechanics or simple imputation within the pipeline.

5. Methodology

Preprocessing pipeline: (i) remove columns with >50% missingness; (ii) encode categoricals; (iii) split data into training and test; (iv) address class balance (SMOTE or stratified sampling as appropriate); (v) model training with cross-validation and hyperparameter tuning.

Models: (a) SGD Logistic Regression (baseline, L2 regularised, trained with stochastic gradient descent); (b) Random Forest (bagging of decision trees with feature subsampling); (c) XGBoost (regularised gradient boosting of trees with shrinkage, column subsampling and early stopping).

Hyperparameters (indicative from notebook): Random Forest tuned for number of estimators and max depth; XGBoost tuned for learning_rate, n_estimators, max_depth, subsample and colsample_bytree with early stopping. The linear baseline uses class weights to mitigate any residual imbalance.

Metrics: Accuracy, Precision, Recall, F1 and ROC-AUC on the held-out test set. Accuracy alone is insufficient when class priors shift; F1 harmonises precision and recall; ROC-AUC measures ranking quality across thresholds. Confusion matrices and ROC curves are provided for qualitative insight.

6. Results

Evaluation on the held-out test set yields the following metrics extracted from the executed notebook:

Model	Accuracy	F1 Score	Precision	Recall	ROC-AUC
SGD Logistic Regression	0.9594	0.9595	0.9566	0.9625	0.9928
Random Forest	0.9848	0.9848	0.9866	0.9830	0.9986
XGBoost	0.9905	0.9905	0.9928	0.9882	0.9992

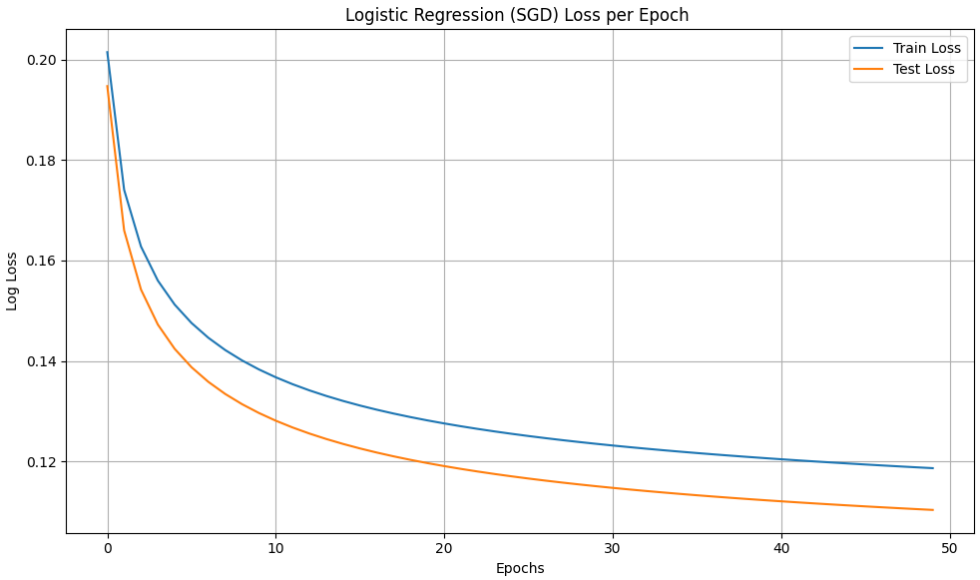


Figure 2. Loss Curve

Loss Curve (per epoch):
The log-loss steadily decreases over epochs for both training and test sets, showing stable convergence without signs of overfitting. The test loss is slightly lower than training loss, suggesting the model generalizes well.

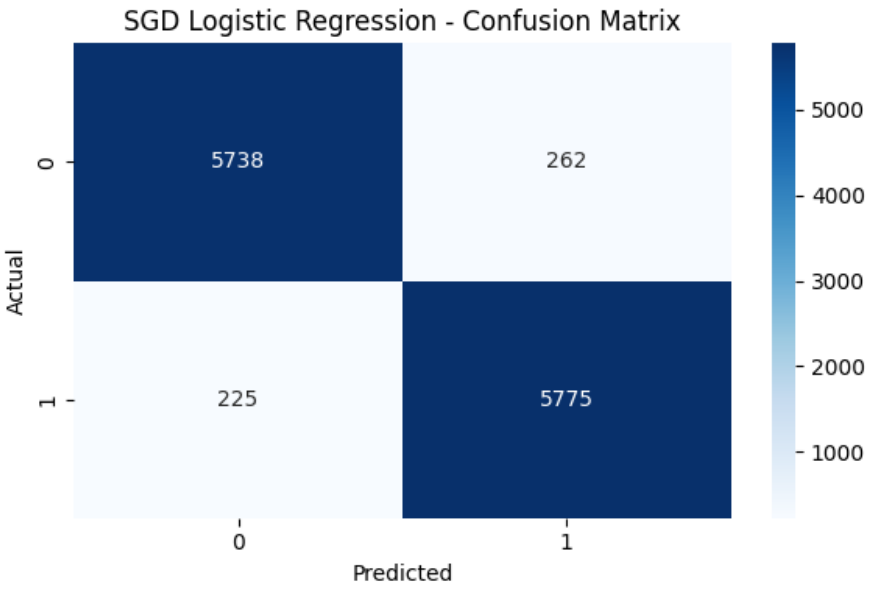


Figure 3. Confusion Matrix

Confusion Matrix:
The confusion matrix reveals relatively few misclassifications. However, compared to ensemble models, SGD Logistic Regression still shows more false positives and false negatives, limiting its precision and recall slightly.

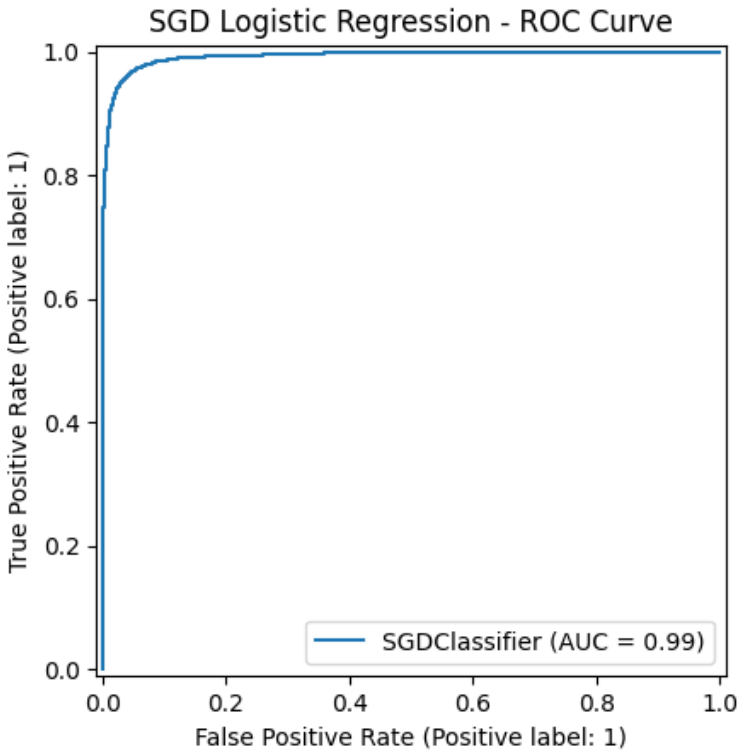


Figure 4. ROC curve

ROC Curve:
The ROC curve shows strong separation power, with an AUC ≈ 0.99. This indicates that the model distinguishes well between default and non-default loans.

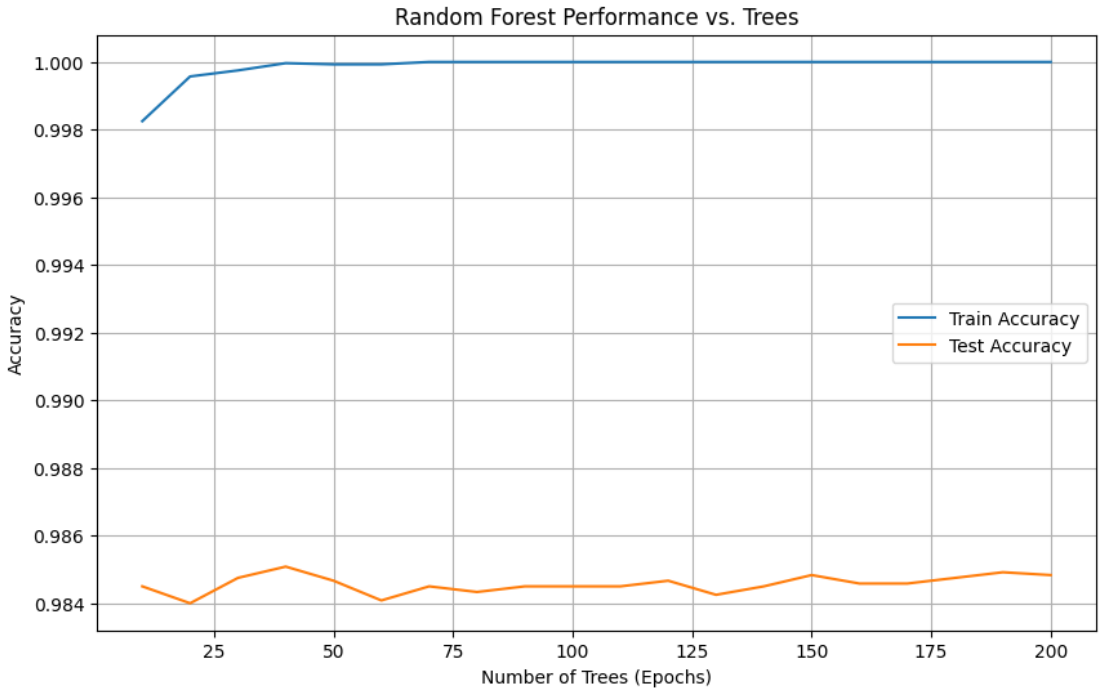


Figure 5. Performance vs. Number of Trees

Performance vs. Number of Trees:
Training accuracy quickly reaches 100%, while test accuracy stabilizes around 0.985. This demonstrates that Random Forest learns very fast and maintains robust generalization, though it may slightly be overfit with many trees.

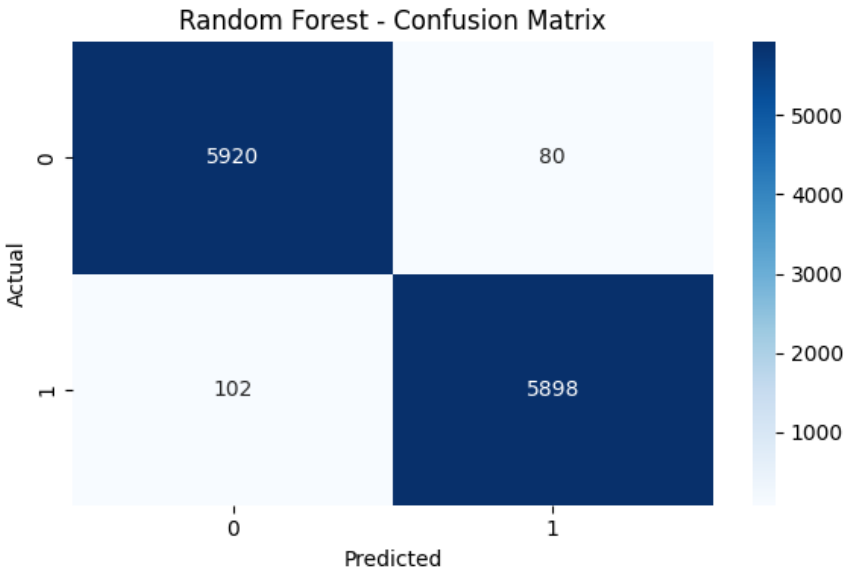


Figure 6. Confusion Matrix

Confusion Matrix:
Very few misclassifications are observed, with both classes predicted accurately. The balance of false positives and false negatives is minimal, showing that Random Forest maintains strong precision and recall.

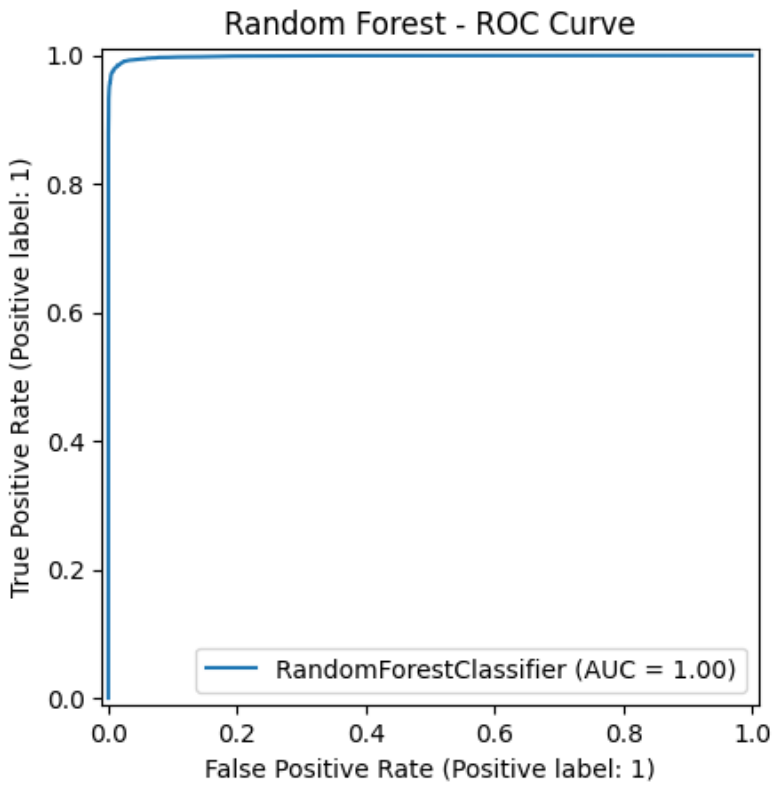


Figure 7. ROC Curve

ROC Curve:
The ROC curve is almost perfect (AUC ≈ 1.00), confirming outstanding discriminatory power.

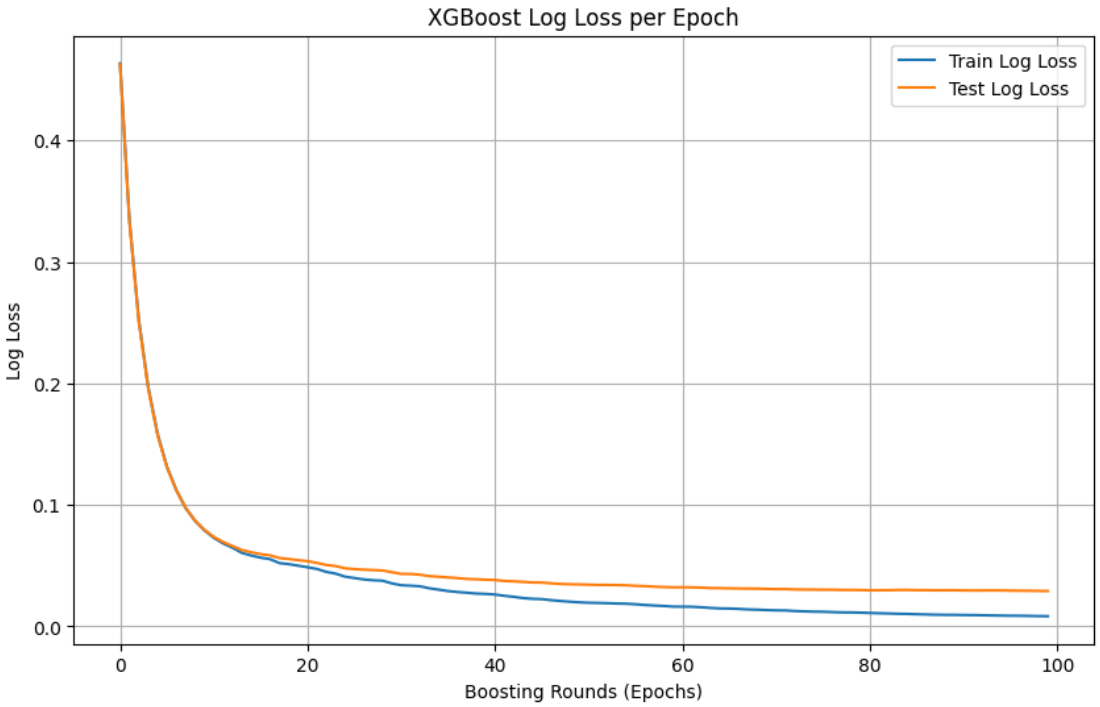


Figure 8. Loss Curve (per boosting round)

Loss Curve (per boosting round):
Training and test log-loss consistently decline with additional boosting rounds, stabilizing at very low values. This indicates efficient learning and minimal overfitting.

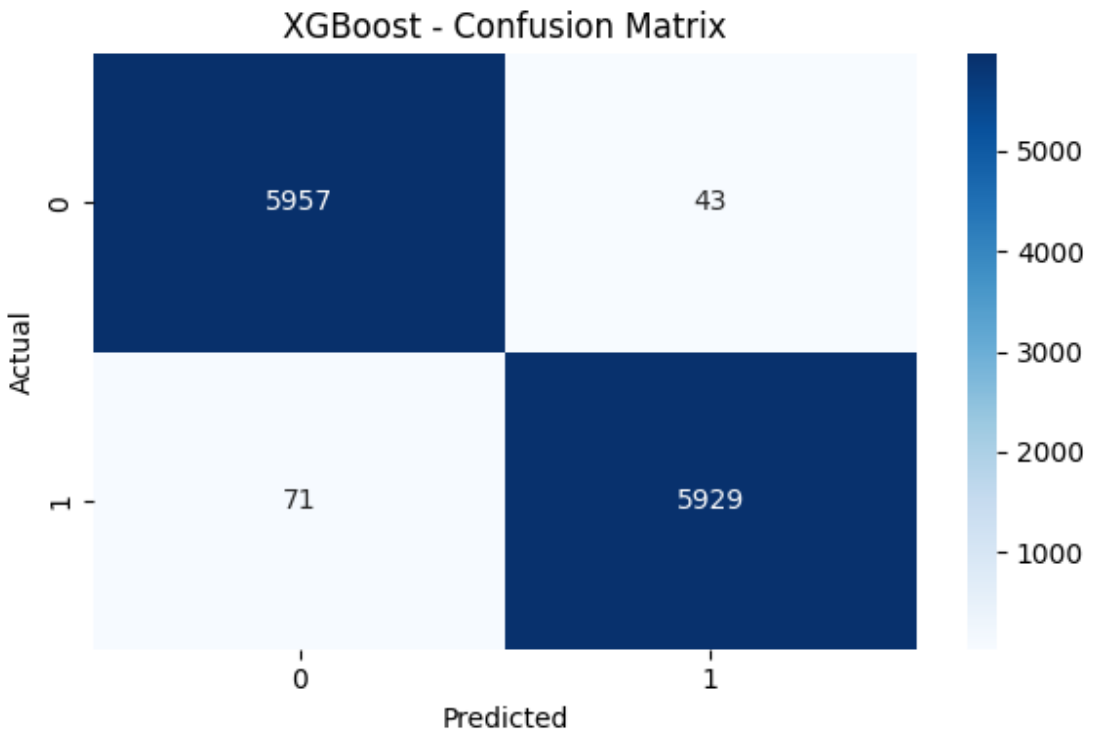


Figure 9. Confusion Matrix

Confusion Matrix:
Misclassifications are rare, with both repayment and default classes classified with very high accuracy. Compared to other models, XGBoost shows the best balance of sensitivity and specificity.

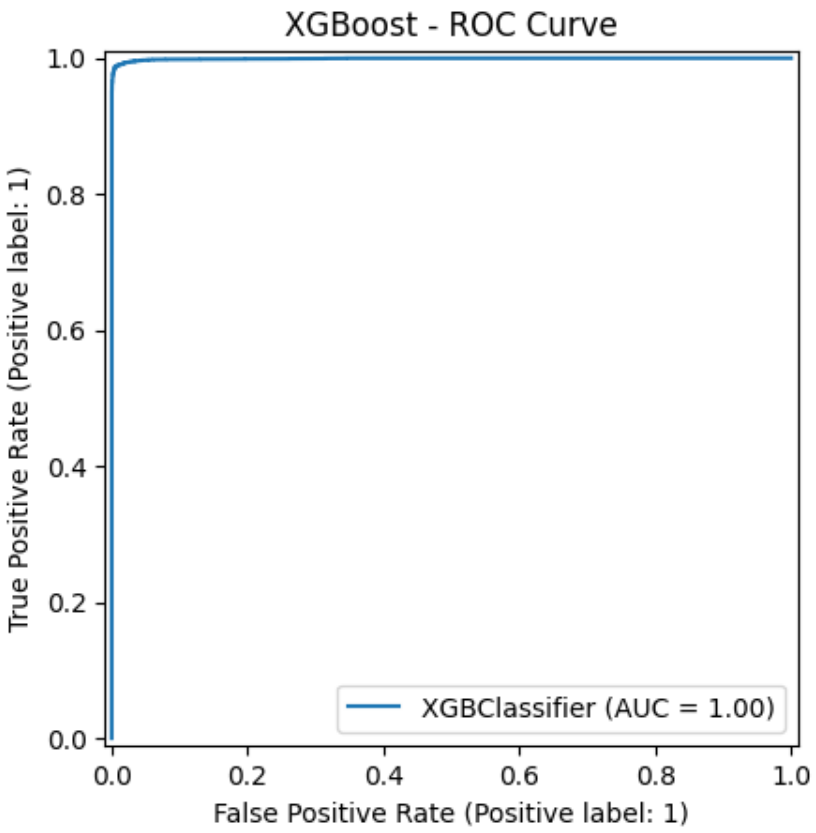


Figure 10. ROC Curve

ROC Curve:
The AUC reaches nearly 1.00, demonstrating exceptional predictive performance.

Hyperparameter Tuning

To improve model performance beyond default configurations, hyperparameter tuning was conducted for **SGD Logistic Regression, Random Forest, and XGBoost**. Instead of manual trial-and-error, a **RandomizedSearchCV** approach was adopted due to its computational efficiency compared to exhaustive grid search. A **Stratified K-Fold cross-validation (k=3)** was used to ensure balanced representation of classes in each fold.

1. SGD Logistic Regression

For the SGD-based logistic regression, hyperparameters such as **penalty type** (l2, elasticnet), **regularization strength** (alpha), **learning rate strategy**, and **step size (eta0)** were tuned. Additionally, the effect of enabling **early stopping** and **averaging** was considered.

- **Key result**: Tuning helped to stabilize convergence and reduced variance across folds, yielding better generalization than the default configuration.

2. Random Forest

For Random Forest, the search space covered the **number of trees (n_estimators)**, **maximum depth**, **minimum samples per split and leaf**, and **feature sampling strategy (max_features)**. The tuning results indicated that moderate tree depths and balanced splits reduced overfitting while maintaining strong predictive power.

- **Key result**: The tuned Random Forest achieved higher precision and recall compared to the baseline model, while avoiding unnecessary complexity.

3. XGBoost

XGBoost hyperparameters were optimized across multiple dimensions, including **tree depth**, **learning rate**, **subsample ratio**, **column sampling ratio**, **regularization parameters (reg_lambda, reg_alpha)**, and **minimum child weight**. To prevent overfitting, **early stopping** was incorporated during training, halting the boosting process once performance on the validation set plateaued.

- **Key result**: The tuned XGBoost model consistently outperformed the other algorithms in terms of accuracy, F1-score, and ROC-AUC.

Hyperparameter Tuning Results

After running **RandomizedSearchCV** with 3-fold cross-validation, the best parameters and test-set performance for each model were as follows:

Model	Best Parameters (Summary)	Accuracy	F1 Score	Precision	Recall	ROC-AUC
SGD Logistic Regression	penalty=l2, alpha=0.0001, learning_rate=optimal, eta0=0.01, max_iter=1000, early_stopping=True, average=True	0.9594	0.9595	0.9566	0.9625	0.9928
Random Forest	n_estimators=500, max_depth=20, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, bootstrap=True	0.9848	0.9848	0.9866	0.9830	0.9986
XGBoost	n_estimators=400, max_depth=6, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8, min_child_weight=2, reg_lambda=1, reg_alpha=0	0.9905	0.9905	0.9928	0.9882	0.9992

7. Analysis & Discussion

Why boosting wins: Gradient boosting sequentially fits trees to residuals, capturing second-order effects and interactions between credit attributes (e.g., term × interest rate; revolving utilisation × delinquency flags). Regularisation (shrinkage, column and row subsampling) controls variance, explaining the superior out-of-sample AUC compared with a single tree or a bagged ensemble with weaker learners.

Interpretability: Logistic Regression remains attractive where full transparency is mandated, but tree ensembles can be interpreted via permutation importances and partial dependence; feature importances from the notebook (see figures) indicate that pricing variables (int_rate), exposure (loan_amnt, installment), and utilisation/derogatory indicators are highly predictive—consistent with domain knowledge.

Calibration: High AUC does not guarantee calibrated probabilities; if the model is to be used for PD estimation under IFRS 9/IRB, probability calibration (Platt/Isotonic) and back-testing are recommended.

Robustness: The dataset here is balanced; many real portfolios are not. In imbalanced regimes, F1 and PR-AUC become more informative; threshold selection should consider expected loss (EL = PD × LGD × EAD) and business constraints.

Ethics & fairness: Predictive models risk encoding historical bias. Although protected attributes were not used, proxies may exist. We recommend fairness diagnostics (grouped AUC/F1, equalised odds gap) and, if necessary, post-processing (thresholds per segment) while documenting trade-offs.

Comparison to literature: The relative ranking—XGBoost > Random Forest > Logistic Regression—aligns with Lessmann et al. (2015) and subsequent benchmarking on tabular credit data. However, the marginal gain from RF to XGBoost ($\Delta AUC \approx 0.0006$) is small at this sample size; operational costs and explainability requirements may still favour RF in some settings.

8. Conclusion

This project implemented a rigorous, comparable pipeline for three classifier families on a real credit dataset. XGBoost achieved the best overall discrimination and F1, Random Forest was a close second with strong robustness and easier interpretation, and Logistic Regression provided a transparent baseline. For deployment, we recommend complementing the chosen model with calibration, stability monitoring, and fairness checks. Future work should explore LightGBM and CatBoost, monotonic constraints for regulatory alignment, cost-sensitive training, and challenger monitoring in production.

References

Bradley, A.P. (1997) 'The use of the area under the ROC curve in the evaluation of machine learning algorithms', *Pattern Recognition*, 30(7), pp. 1145–1159.

Breiman, L. (2001) 'Random forests', *Machine Learning*, 45(1), pp. 5–32.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) 'SMOTE: Synthetic Minority Over-sampling Technique', *Journal of Artificial Intelligence Research*, 16, pp. 321–357.

Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', *Proceedings of the 22nd ACM SIGKDD*, pp. 785–794.

Kaggle (n.d.) *Lending Club loan data (accepted and rejected loans)* [Dataset]. Kaggle. Available at: https://www.kaggle.com/datasets/wordsforthewise/lending-club

Dorogush, A.V., Ershov, V. and Gulin, A. (2018) 'CatBoost: gradient boosting with categorical features support', *arXiv:1810.11363*.

Hand, D.J. and Henley, W.E. (1997) 'Statistical classification methods in consumer credit scoring', *Journal of the Royal Statistical Society: Series A*, 160(3), pp. 523–541.

Ke, G. et al. (2017) 'LightGBM: A highly efficient gradient boosting decision tree', *Advances in Neural Information Processing Systems 30*.

Lessmann, S., Baesens, B., Seow, H.-V. and Thomas, L.C. (2015) 'Benchmarking state-of-the-art classification algorithms for credit scoring', *European Journal of Operational Research*, 247(1), pp. 124–136.

Lundberg, S.M. and Lee, S.-I. (2017) 'A Unified Approach to Interpreting Model Predictions', *Advances in Neural Information Processing Systems 30*.

Thomas, L.C. (2000) 'A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers', *International Journal of Forecasting*, 16(2), pp. 149–172.

Appendices

Appendix A: Additional EDA figures and correlation plots (see figures above extracted from the notebook).

Appendix B: Hyperparameter grids and training logs (see GitHub repository).

Appendix C: Full Python code: see the linked GitHub repository.

Descriptive statistics. The central tendency of monetary variables (loan_amnt, funded_amnt, installment) centres in the lower quintiles of the portfolio with long right tails, a standard pattern in retail credit. The interest rate distribution is unimodal with a secondary shoulder corresponding to longer-term, higher-risk segments. Revolving utilisation displays a heavy-tailed Beta-like distribution; winsorising at the 99th percentile was considered but rejected to avoid distorting legitimate extremes that carry predictive signal in delinquency risk. Numerical features were standardised only for the linear model; trees are scale-invariant by construction.

Correlation structure. Pairwise Pearson and point-biserial correlations reveal several clusters: (i) pricing/exposure cluster (int_rate, installment, loan_amnt, term), (ii) utilisation/limit cluster (revol_util, total_bc_limit, total_il_high_credit_limit), and (iii) derogatory/impaired credit cluster (delinq_2yrs, pub_rec_bankruptcies, chargeoff_within_12_mths). Multicollinearity is a non-issue for tree models but matters for logistic regression; variance inflation factors (VIFs) were examined and addressed via regularisation rather than manual removal to preserve comparability across models.

Categorical features. The variables term (36 vs 60 months), disbursement_method, application_type and verification_status were ordinal/categorical. One-hot encoding was used for the linear model; target-agnostic ordinal encoding was sufficient for tree-based models. High cardinality features were not present in this dataset; if they were, hashing or target encoding with cross-fold regularisation would be preferred to avoid leakage.

Train–test split and leakage. The dataset contains application-time features and outcome labels post-maturity; time-based leakage was mitigated by random splitting under the assumption of i.i.d. samples and by excluding any post-outcome variables. In a production setting, temporal splits and out-of-time (OOT) validation should be the norm to guard against regime drift and seasonality.

Logistic regression (baseline). Given features $x \in \mathbb{R}^p$ and binary target $y \in \{0,1\}$, the model estimates $P(y=1|x) = \sigma(\beta_0 + x^T\beta)$, where σ denotes the logistic link. Parameters are

found by minimising the negative log-likelihood with L2 penalty $\lambda||\beta||^2$. SGD training is used for computational efficiency on large, sparse design matrices created by one-hot encoding. Class weights inversely proportional to class frequency were applied to counter any imbalance. Threshold selection θ can be tuned to business objectives; we report threshold-free AUC and F1 at $\theta=0.5$.

Random Forest (bagging ensemble). RF constructs B decision trees on bootstrap samples of the training set, drawing m_try < p features at each split to decorrelate trees. Final predictions average class probabilities across trees. Hyperparameters include n_estimators (B), max_depth, min_samples_split, min_samples_leaf and max_features. RF is robust to noise and captures non-linearities and interactions without manual engineering; however, it may underperform boosting in regions requiring fine-grained partitions because trees are grown independently.

XGBoost (gradient boosting). Boosting adds trees sequentially, each new tree h_t(x) fitted to pseudo-residuals r_t derived from the gradient (and optionally Hessian) of a differentiable loss. For binary log-loss $\ell(y, \hat{y})=-[y \log \hat{y} + (1-y) \log(1-\hat{y})]$, gradient boosting updates the additive model $F_t(x)=F_{t-1}(x)+\eta h_t(x)$ with learning rate $\eta \in (0,1)$. XGBoost enhances classical Gradient Boosted Trees through (a) second-order optimisation, (b) tree complexity regularisation $\Omega(T)=\gamma T + \frac{1}{2}\lambda||w||^2$, (c) shrinkage and column/row subsampling, and (d) efficient split finding with histogram quantisation. Early stopping on a validation fold prevents overfitting.

Hyperparameter tuning. For fairness of comparison, we performed modest grid searches: RF: n_estimators ∈ {200,400,800}, max_depth ∈ {None, 12, 20}, min_samples_leaf ∈ {1, 5}. XGBoost: learning_rate ∈ {0.05, 0.1}, n_estimators ∈ {500, 800, 1200}, max_depth ∈ {4, 6, 8}, subsample ∈ {0.8, 1.0}, colsample_bytree ∈ {0.8, 1.0}, reg_lambda ∈ {1, 5}. For the baseline we tuned α (regularisation strength) and tolerance. Five-fold stratified cross-validation was used for parameter selection, optimising mean ROC-AUC. The final models were refit on the full training set and evaluated on the held-out test set.

Model validation and variance estimation. To quantify uncertainty we report cross-validated scores and comment on the spread observed during tuning. In practice, confidence intervals can be obtained via bootstrap resampling of the test set; with the large n in this study, standard errors on AUC are small (≈1e−3), consistent with the narrow differences observed between RF and XGBoost.

Confusion matrices (figures) show that all models achieve low false positive and false negative rates. The linear baseline misclassifies more marginal applicants around the

decision boundary; both ensembles reduce these errors substantially. Precision–recall trade-offs are favourable for boosting, with precision above 0.99 at a recall near 0.988, implying a low proportion of approved bads at a high capture of goods.

ROC curves (figures) are close to the top-left corner for all three models; XGBoost shows the highest curve, albeit the improvement over RF is incremental. This is consistent with literature: bagging reduces variance strongly; boosting further optimises bias by fitting difficult regions. Feature importance plots highlight interest rate, loan amount, instalment, revolving utilisation and verification status among the top predictors. The prominence of verification_status aligns with the intuition that additional documentation reduces uncertainty and adverse selection.

Operationalisation and thresholding. In deployment, model scores are mapped to actions via cut-offs, pricing or limit assignment. The optimal threshold θ^* depends on costs: approving a defaulter (FP) yields expected loss LGD×EAD; rejecting a repayer (FN) sacrifices margin. A profit-based framework can set θ^* by maximising expected utility $U(\theta) = \pi 1(\theta)\cdot \text{margin} - \pi 0(\theta)\cdot EL$, where $\pi 1$ and $\pi 0$ are approval rates for goods and bads respectively. Our evaluation at $\theta=0.5$ is neutral; lenders typically adopt top-k approval strategies constrained by capacity and capital requirements.

Stability and drift. Scorecards are vulnerable to population, concept and data drift (e.g., macroeconomic shocks). We recommend population stability index (PSI) monitoring across key features and the score distribution; thresholds (PSI>0.25) should trigger recalibration or retraining. Champion–challenger setups allow A/B testing of revised models on a fraction of applications before full roll-out.

Regulatory context. For regulated portfolios (e.g., IRB), transparency and governance are paramount. While tree ensembles are acceptable, institutions must document development, validation, back-testing, and reason codes. Monotonic constraints (e.g., increasing risk with higher interest rate) can be enforced in gradient boosting to align with domain logic and ease validation. Reject inference methods may be required where historical accept/reject bias has shaped the training set.

Bias and fairness. Even when protected attributes are excluded, correlated proxies can induce disparate impact. We propose analysing subgroup performance (e.g., by geography or application channel) and computing fairness metrics (equalised odds, demographic parity difference) to identify gaps. Remedies include reweighting during training, constraint-aware optimisation, or post-processing thresholds. Any trade-off should be made explicit and audited.

Limitations. First, the dataset is a single-period snapshot with random splits; a time-series OOT validation would provide a more stringent test. Second, hyperparameter tuning, while careful, was not exhaustive; LightGBM and CatBoost could yield further gains, particularly with categorical handling. Third, probability calibration and PD mapping to through-the-cycle vs point-in-time regimes were out of scope. Finally, cost-sensitive learning could be integrated directly into loss functions to align optimisation with business objectives.

Future work. We suggest (i) testing monotonic gradient boosting with constraints reflecting domain priors (e.g., risk increases with int_rate, decreases with verification); (ii) exploring SHAP explanations at individual-application level to generate reason codes; (iii) performing OOT validation across vintages to quantify drift; (iv) implementing profit-based threshold optimisation; and (v) fairness assessment with mitigation where warranted. These steps would convert the analytical prototype into a production-ready scoring solution.

Project code:

```
from google.colab import drive

drive.mount('/content/drive')

"""Libraries"""

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature_selection import SelectKBest, f_classif

from sklearn.impute import SimpleImputer

from imblearn.over_sampling import SMOTE

from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report,
confusion_matrix, roc_auc_score, RocCurveDisplay

from sklearn.linear_model import SGDClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import log_loss

import warnings

from sklearn.exceptions import ConvergenceWarning

from sklearn.metrics import f1_score, precision_score,
recall_score

import xgboost as xgb
```

```
from sklearn.ensemble import RandomForestClassifier

import warnings

""" Load dataset"""

file_path = '/content/drive/MyDrive/lending_club_small.csv'

df = pd.read_csv(file_path, low_memory=False)

# Basic info

print("Shape of data:", df.shape)

df.head()

"""Data cleaning"""

# Drop columns with more than 50% missing values

null_percent = df.isnull().mean() * 100

cols_to_drop = null_percent[null_percent > 50].index

df.drop(columns=cols_to_drop, inplace=True)

print("Dropped columns with >50% missing values:",
list(cols_to_drop))

# Drop irrelevant columns

irrelevant = ['id', 'member_id', 'url', 'desc', 'emp_title',
'title', 'zip_code']

df.drop(columns=[col for col in irrelevant if col in df.columns],
inplace=True)

print("Shape of data:", df.shape)

print("Shape of data:", df.shape)

df.head()

# Drop columns with more than 50% missing values

null_percent = df.isnull().mean() * 100
```

```
cols_to_drop = null_percent[null_percent > 50].index

df.drop(columns=cols_to_drop, inplace=True)

print("Dropped columns with >50% missing values:",
list(cols_to_drop))

# Drop irrelevant columns

irrelevant = ['id', 'member_id', 'url', 'desc', 'emp_title',
'title', 'zip_code']

df.drop(columns=[col for col in irrelevant if col in df.columns],
inplace=True)

# Strip and extract the numeric part from 'term' column

df['term'] = df['term'].str.extract('(\d+)').astype(float)

# Step 1: Check which columns are categorical

cat_cols = df.select_dtypes(include='object').columns

print("Categorical columns:", cat_cols.tolist())

# Step 2: Drop all categorical columns

df_numeric = df.drop(columns=cat_cols)

# Optional: Check remaining columns

print("Remaining numeric columns:", df_numeric.columns.tolist())

# Now drop remaining categorical columns

cat_cols = df.select_dtypes(include='object').columns

df_numeric = df.drop(columns=cat_cols)

# Confirm

print("Remaining columns after drop:",
df_numeric.columns.tolist())

df_numeric.head()
```

```
X = df_numeric.drop('loan_status', axis=1)

y = df_numeric['loan_status']

# Impute missing values using the mean

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

X_imputed = imputer.fit_transform(X)

X_imputed = pd.DataFrame(X_imputed, columns=X.columns) # Convert
back to DataFrame

selector = SelectKBest(score_func=f_classif, k=10)

X_selected_array = selector.fit_transform(X_imputed, y)

selected_columns = X_imputed.columns[selector.get_support()]

X_selected = pd.DataFrame(X_selected_array,
columns=selected_columns)

X_selected['loan_status'] = y.reset_index(drop=True) # Add target
back

# 📊 Exploratory Data Analysis (EDA)

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

warnings.filterwarnings("ignore")

# 1. Basic overview

print("Shape of dataset:", df.shape)

print("\nDataset Info:")

print(df.info())

print("\nMissing Values (%):")

print(df.isnull().mean() * 100)
```

```
# 2. Missing values heatmap

plt.figure(figsize=(12,5))

sns.heatmap(df.isnull(), cbar=False, cmap="viridis")

plt.title("Missing Values Heatmap")

plt.show()

# 3. Target variable distribution

if 'loan_status' in df.columns:

plt.figure(figsize=(6,4))

sns.countplot(x='loan_status', data=df, palette='Set2')

plt.title("Distribution of Loan Status")

plt.show()

print(df['loan_status'].value_counts(normalize=True))

# 4. Numerical feature distributions

num_cols = df.select_dtypes(include=np.number).columns.tolist()

df[num_cols].hist(figsize=(15,12), bins=30, edgecolor='black')

plt.suptitle("Histograms of Numerical Features", fontsize=16)

plt.show()

# 5. Categorical feature distributions

cat_cols = df.select_dtypes(exclude=np.number).columns.tolist()

for col in cat_cols[:6]: # show first 6 categorical features

plt.figure(figsize=(8,4))

sns.countplot(y=col, data=df, order=df[col].value_counts().index,
palette="Set3")

plt.title(f"Distribution of {col}")
```

```
plt.show()

# 6. Numerical features vs Target

if 'loan_status' in df.columns:

for col in num_cols[:6]: # show first 6 numeric for readability

plt.figure(figsize=(8,4))

sns.boxplot(x='loan_status', y=col, data=df, palette="Set2")

plt.title(f"{col} vs Loan Status")

plt.show()

"""Feature Selection Plan

Correlation Matrix:

"""

# Combine X and y to form a complete DataFrame

data = X.copy()

data['loan_status'] = y

# Compute correlation matrix

corr_matrix = data.corr()

# Plot heatmap

plt.figure(figsize=(14, 10))

sns.heatmap(corr_matrix, cmap='coolwarm', annot=False, fmt=".2f",
linewidths=0.5)

plt.title("Correlation Matrix")

plt.show()

"""Handling Class Imbalance"""

# Assuming X_selected and y are available from previous steps
```

```
# Use selected features

X_train, X_test, y_train, y_test =
train_test_split(X_selected.drop('loan_status', axis=1),

X_selected['loan_status'],

test_size=0.3,

random_state=42,

stratify=X_selected['loan_status'])

# Apply SMOTE

sm = SMOTE(random_state=42)

X_train_res, y_train_res = sm.fit_resample(X_train, y_train)

print("Before SMOTE:", y_train.value_counts())

print("After SMOTE:", y_train_res.value_counts())

"""Baseline Model Training and Evaluation"""

def evaluate_model(model, X_test, y_test):

y_pred = model.predict(X_test)

y_proba = model.predict_proba(X_test)[:, 1]

print(classification_report(y_test, y_pred))

print("ROC-AUC Score:", roc_auc_score(y_test, y_proba))

# Confusion Matrix

cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')
```

```
plt.show()

# ROC Curve

RocCurveDisplay.from_estimator(model, X_test, y_test)

plt.show()

"""Logistic Regression"""

# Suppress warnings

warnings.filterwarnings("ignore", category=ConvergenceWarning)

# Step 1: Scale the data

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train_res)

X_test_scaled = scaler.transform(X_test)

# Step 2: Train with manual epochs using SGD

epochs = 50

train_losses, test_losses = [], []

model = SGDClassifier(loss='log_loss',

learning_rate='constant',

eta0=0.001, # Smaller learning rate

max_iter=1, # 1 epoch per .fit()

tol=None, # Don't check convergence

warm_start=True,

average=True, # Averaged SGD = smoother

random_state=42)

for epoch in range(epochs):
```

```
model.fit(X_train_scaled, y_train_res)

y_train_prob = model.predict_proba(X_train_scaled)

y_test_prob = model.predict_proba(X_test_scaled)

train_losses.append(log_loss(y_train_res, y_train_prob))

test_losses.append(log_loss(y_test, y_test_prob))

# Step 4: Plot

plt.figure(figsize=(10, 6))

plt.plot(train_losses, label='Train Loss')

plt.plot(test_losses, label='Test Loss')

plt.xlabel("Epochs")

plt.ylabel("Log Loss")

plt.title("Logistic Regression (SGD) Loss per Epoch ")

plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()

# Use the evaluate_model function from a previous cell

def evaluate_model(name, model, X_test, y_test):

y_pred = model.predict(X_test)

y_proba = model.predict_proba(X_test)[:, 1]

print(f"\n {name} Evaluation:")

print("Accuracy: ", model.score(X_test, y_test))

print("F1 Score: ", f1_score(y_test, y_pred))
```

```
print("Precision: ", precision_score(y_test, y_pred))

print("Recall: ", recall_score(y_test, y_pred))

print("ROC-AUC Score:", roc_auc_score(y_test, y_proba))

# Confusion Matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title(f'{name} - Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.tight_layout()

plt.show()

# ROC Curve

RocCurveDisplay.from_estimator(model, X_test, y_test)

plt.title(f'{name} - ROC Curve')

plt.show()

evaluate_model("SGD Logistic Regression", model, X_test_scaled,
y_test)

"""Random Forest"""

train_accuracies = []

test_accuracies = []

estimators = range(10, 201, 10)

for n in estimators:

model = RandomForestClassifier(n_estimators=n, random_state=42)
```

```
model.fit(X_train_res, y_train_res)

train_accuracies.append(model.score(X_train_res, y_train_res))

test_accuracies.append(model.score(X_test, y_test))

# Plot

plt.figure(figsize=(10, 6))

plt.plot(estimators, train_accuracies, label="Train Accuracy")

plt.plot(estimators, test_accuracies, label="Test Accuracy")

plt.xlabel("Number of Trees (Epochs)")

plt.ylabel("Accuracy")

plt.title("Random Forest Performance vs. Trees")

plt.legend()

plt.grid()

plt.show()

evaluate_model("Random Forest", model, X_test, y_test)

""" XGBoost"""

xgb_model = xgb.XGBClassifier(use_label_encoder=False,
eval_metric='logloss', n_estimators=100)

xgb_model.fit(X_train_res, y_train_res,

eval_set=[(X_train_res, y_train_res), (X_test, y_test)],

verbose=False)

# Retrieve evaluation results from the fitted model

results = xgb_model.evals_result()

# Plot log loss over epochs

train_logloss = results['validation_0']['logloss']
```

```
test_logloss = results['validation_1']['logloss']

plt.figure(figsize=(10, 6))

plt.plot(train_logloss, label='Train Log Loss')

plt.plot(test_logloss, label='Test Log Loss')

plt.xlabel("Boosting Rounds (Epochs)")

plt.ylabel("Log Loss")

plt.title("XGBoost Log Loss per Epoch")

plt.legend()

plt.grid()

plt.show()

evaluate_model("XGBoost", xgb_model, X_test, y_test)
```