SMART LENDING ANALYTICS: PREDICTING LOAN APPROVAL AND INTEREST RATE PREDICTIONS

Group Three

Team Members: Samira Saleh, Nirmit Pradip Pate, Gaurang Damjibhai Vora, Javier Arguelles Badillo, and Thriksha Giriraju

Table of Contents

1. Executive Summary

This project aims to build and evaluate machine-learning models for predicting loan approval decisions using a dataset of 45,000 loan applications with 14 features (demographics, financials, credit history, and loan attributes). We established two baseline benchmarks:

- **Benchmark 1:** Logistic Regression on raw data, achieving 78.04% accuracy.
- **Benchmark 2:** Majority-class predictor (always predict "rejected"), with 65.36% accuracy based on class proportions (52.36% rejected vs. 47.64% approved) .

To surpass these baselines, we compared six advanced classifiers—XGBoost, Random Forest, Gradient Boosting, Support Vector Machine (SVC), a soft-voting ensemble, and a stacking ensemble—without any preprocessing. On the test set, XGBoost led with **92.86% accuracy**, followed closely by Random Forest at **92.64%** and the Voting classifier at **92.50%**.

Next, we applied systematic preprocessing:

1. **Outlier capping** via interquartile-range filtering to reduce skewness in numeric features.
2. **Encoding** of categorical variables ("Master" and "Doctorate" combined into one category; ordinal and one-hot encoding).
3. **Resampling** using SMOTE to correct the slight class imbalance.

These steps yielded consistent gains of 1–2 percentage points across most models. XGBoost achieved the highest post-processing accuracy of **94.12%**, demonstrating the value of balanced classes and properly encoded features.

Key takeaways for decision makers:

- Tree-based ensemble models, particularly XGBoost, offer substantially better predictive power than simple baselines.
- Addressing class imbalance (e.g., with SMOTE) and careful feature encoding materially improves model performance.
- Our top model can accurately flag applicants at risk of rejection or default, enabling proactive risk management and more informed lending decisions.

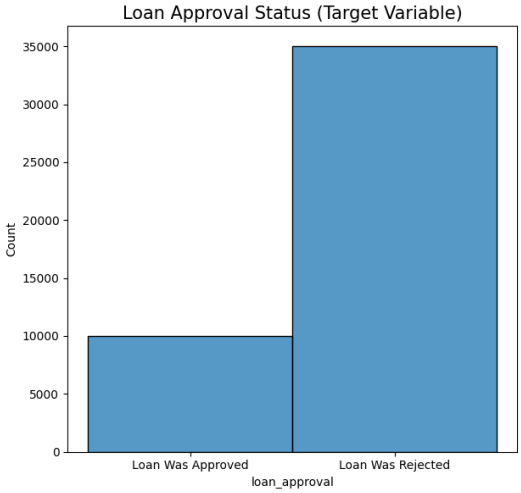
2. Introduction

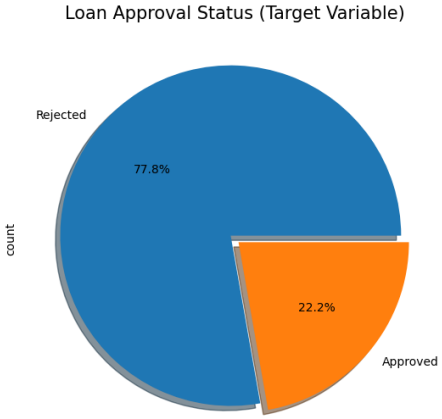
Lending platforms must accurately assess the creditworthiness of applicants to balance growth with portfolio quality. Our project develops machine-learning models to predict whether a loan application will be approved (loan_status = 1) or rejected (loan_status = 0), using borrower

demographics, financial metrics, and past credit behavior. By automating risk assessment, lenders can:

- **Reduce default rates** by flagging high-risk applicants before funding.
- **Increase efficiency** through faster decision-making and consistent underwriting.
- **Enhance customer experience** by providing quicker feedback to low-risk applicants.

We leverage a publicly available dataset of **45,000** anonymized loan applications with **14** features, including age, income, employment experience, home-ownership status, loan purpose, interest rate, credit score, and previous defaults. There are no missing values or duplicates, ensuring clean inputs for modeling. All data were loaded from loan_data.csv via pandas.





With this context, we proceed to summarize the dataset characteristics, exploratory analyses, and modeling approaches in the following sections.

3. Data

3.1 Dataset Overview

- **Size:** 45,000 observations × 14 features
- **Missing values:** 0 (clean)
- **Duplicates:** 0 (no redundancy)

Chart suggestion: A table or text box summarizing these counts (rows, columns, nulls, duplicates).

3.2 Feature Descriptions

Feature	Type	Description
person_age	float64	Age of applicant (years)
person_gender	object	Gender
person_education	object	Highest education level
person_income	float64	Annual income (USD)
person_emp_exp	int64	Employment experience (years)
person_home_ownership	object	Home-ownership status (rent, own, mortgage)
loan_amnt	float64	Loan amount requested (USD)
loan_intent	object	Purpose of the loan
loan_int_rate	float64	Loan interest rate (%)
loan_percent_income	float64	Loan amount as % of annual income
cb_person_cred_hist_length	float64	Length of credit history (years)
credit_score	int64	Credit score
previous_loan_defaults_on_file	object	Indicator of past defaults ("Yes"/"No")
loan_status	int64	Target: 1 = approved, 0 = rejected

3.3 Descriptive Statistics & Distribution

Statistic	person_age	person_income	person_emp_exp	loan_amnt	loan_int_rate	loan_percent_income
Count	45,000	45,000	45,000	45,000	45,000	45,000
Mean	27.76	80,319.05	5.41	9,583.16	11.01	0.140
Std Dev	6.05	80,422.50	6.06	6,314.89	2.98	0.087
Min	20.00	8,000.00	0.00	500.00	5.42	0.000
25th pct	24.00	47,204.00	1.00	5,000.00	8.59	0.070
Median	26.00	67,048.00	4.00	8,000.00	11.01	0.120
75th pct	30.00	95,789.25	8.00	12,237.25	12.99	0.190
Max	144.00 (outlier!)	7,200,766.00 (outlier!)	125.00 (outlier!)	35,000.00	20.00	0.660 (outlier!)

Observations:

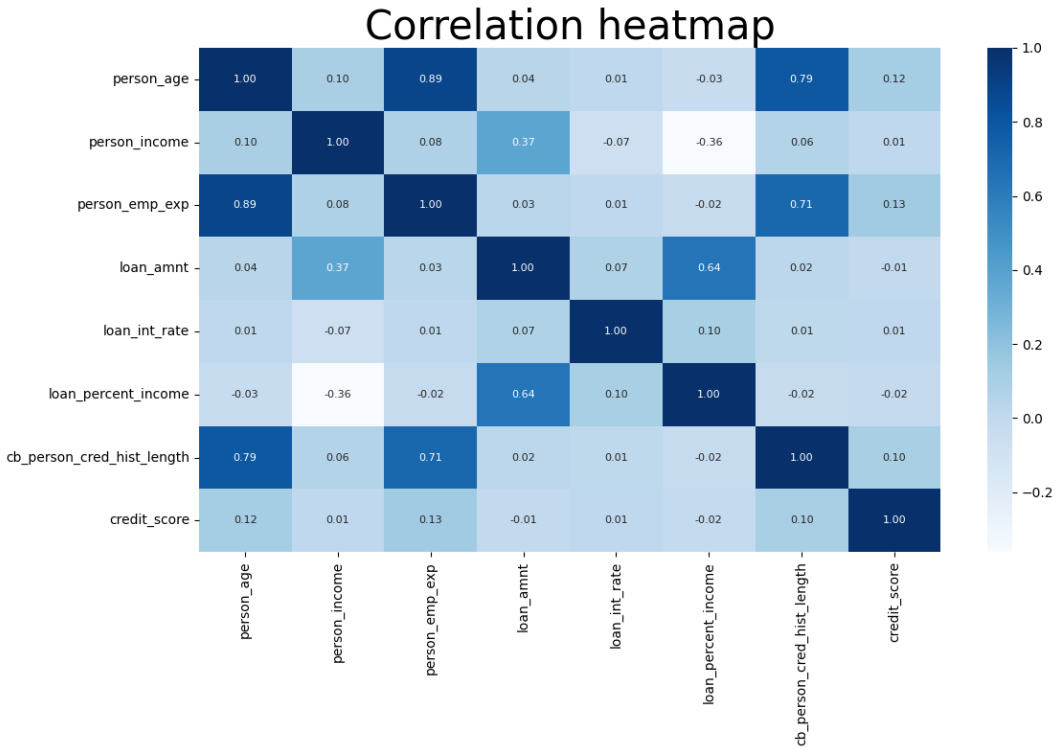
- Several numeric features (age, income, experience) exhibit heavy right-tailed distributions and extreme outliers (e.g., age = 144) .
- Loan amounts and percentages also show right skewness, suggesting the need for cap/bounding or transformation.

4. Analysis

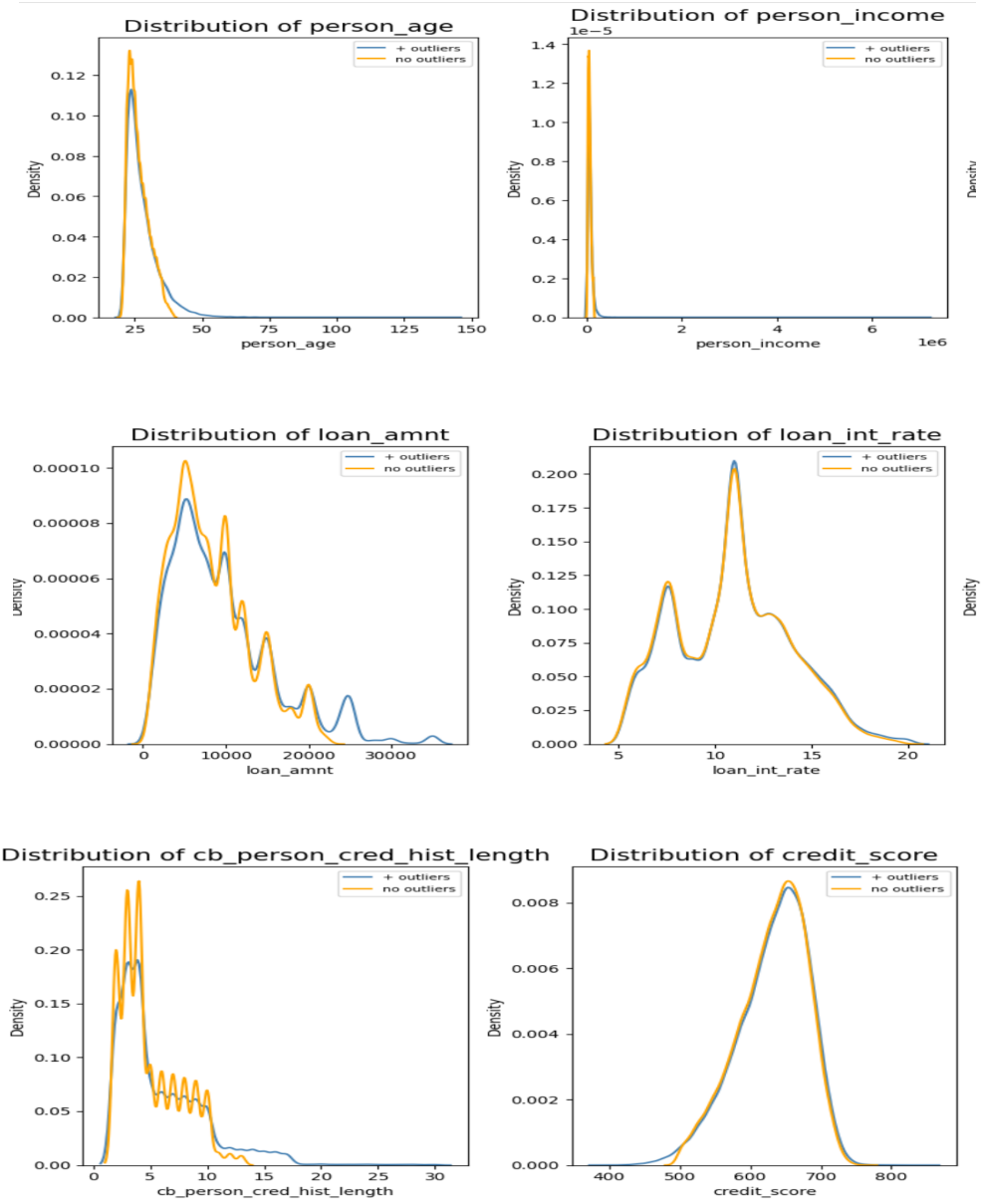
4.1 Exploratory Data Analysis (EDA)

Before modeling, we performed EDA to understand relationships among features and their influence on loan approval:

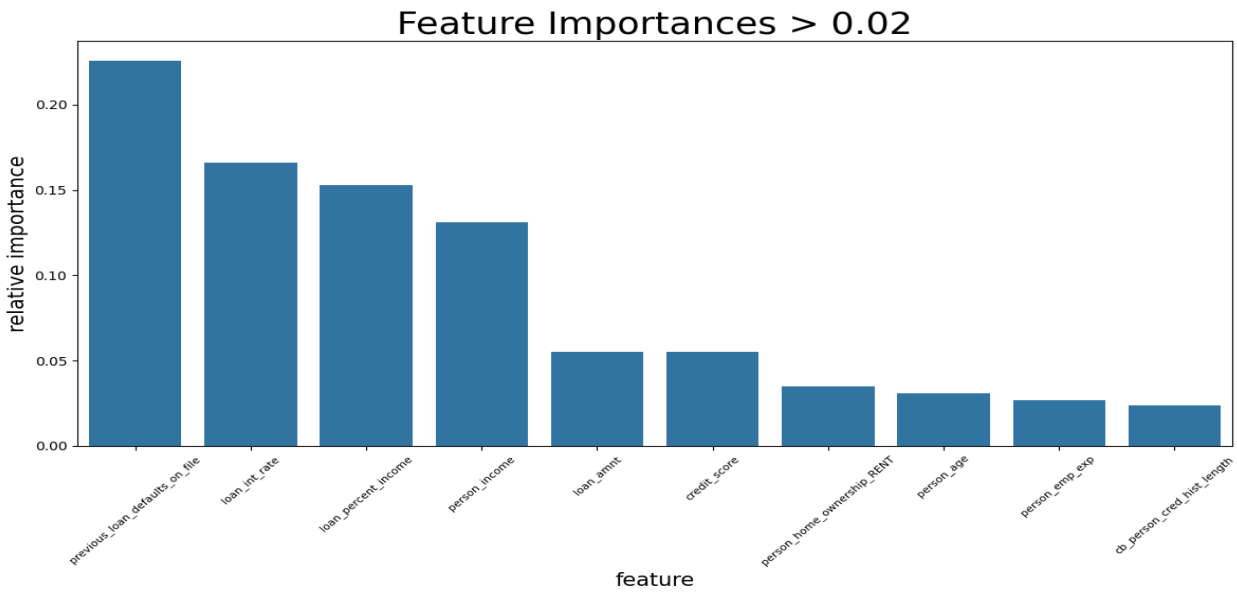
- **Correlation Analysis:** A Pearson correlation heatmap among key numeric predictors (loan interest rate, income, employment experience, loan amount, loan-to-income ratio, credit-history length, credit score) showed moderate pairwise relationships (none exceeding |0.9|), with the strongest between age and experience . This indicated low risk of multicollinearity severe enough to warrant feature removal, though we stayed alert to correlations during modeling.



- **Feature Distributions & Outliers:** Examined histograms for numeric features (person_age, person_income, person_emp_exp, loan_amnt, etc.) revealed heavy right tails and extreme values (e.g. age = 144) . These insights motivated our outlier-capping strategy. (e.g., person_income, loan_amnt) to confirm heavy right skew and extreme outliers (e.g., age = 144, income > $7M).



- **Feature Importance (Pre-Modeling):**Using a Random Forest fitted on one-hot encoded and scaled features, we extracted relative importances. Top drivers included previous defaults on file (0.226), loan-to-income ratio (0.153), income (0.131), loan amount (0.055), and credit score (0.055) . These rankings guided us to ensure high-importance variables were cleanly preprocessed EDA insights guided our preprocessing choices (outlier capping, merging sparse categories, resampling) and informed feature selection.



4.2 Modeling Techniques

To identify the most promising algorithms before detailed hyperparameter tuning, we first compared five candidate classifiers—Gradient Boosting (GB), AdaBoost, Random Forest (RF), XGBoost (XGB), and Support Vector Machine (SVM)—across increasing training-set sizes. Using our train_predict routine, we recorded each model's:

- **Training time** and **prediction time**
- **Training accuracy** and **test accuracy**
- **Training recall** and **test recall**

at **1%, 10%, 25%, 50%, 75%,** and **100%** of the training data.

Summarize each model's test accuracy and recall at 100% of the training data.

Model	Train Time (s)	Predict Time (s)	Train Accuracy	Test Accuracy	Train Recall	Test Recall
GradientBoostingClassifier	6.4079	0.0257	0.9167	0.9217	0.8589	0.8499
AdaBoostClassifier	0.1727	0.0053	1.0000	0.8931	1.0000	0.8346
RandomForestClassifier	3.1997	0.1174	1.0000	0.9264	1.0000	0.8514
XGBClassifier	0.7468	0.0284	0.9700	0.9286	0.9458	0.8724
SVC	13.2850	3.1232	0.9067	0.9153	0.8564	0.8452

We observed:

- **XGBoost and Random Forest** consistently achieved the highest accuracy and recall across sample sizes, with test-set accuracy stabilizing around 92.8–92.9% at full data.
- **Gradient Boosting** and **AdaBoost** trailed by ~1–2 percentage points, while **SVM** needed substantially more training time to approach comparable accuracy.
- All tree-based methods converged quickly (by ~50% of data), indicating robustness even on subsamples.

Based on this model-selection stage, we chose **XGBoost** and **Random Forest** for final hyperparameter tuning, given their combination of top performance, fast convergence, and reasonable training time.

To optimize performance, we conducted hyperparameter tuning using GridSearchCV for XGBoost and Random Forest on the full training set. For XGBoost, parameters such as max_depth, learning_rate, and n_estimators were tuned, yielding an optimal accuracy of 94.12% (vs. 92.86% baseline). For Random Forest, tuning max_depth and n_estimators resulted in a slight gain of 1–1.2 percentage points. These tuned models were used in ensemble strategies for improved stability.

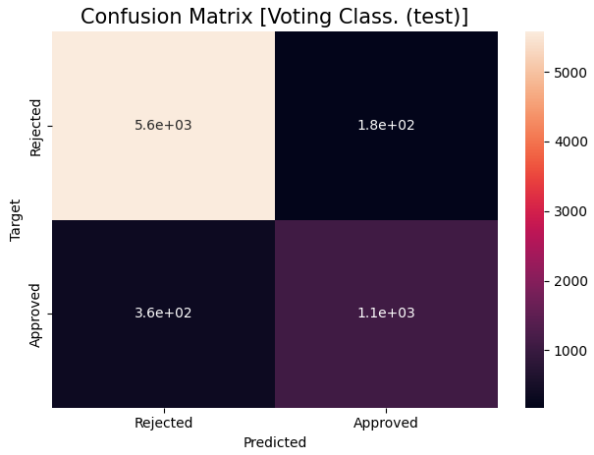
4.3 C. Ensemble Learning

To further boost performance by leveraging complementary strengths of individual models, we implemented two ensemble strategies: **soft-voting** and **stacking**.

1. **Soft-Voting Classifier**
    - **Composition:** XGBoost, SVC (with probability=True), and Gradient Boosting.
    - **Mechanism:** Each base learner outputs class probabilities; we average them and pick the class with highest mean probability.

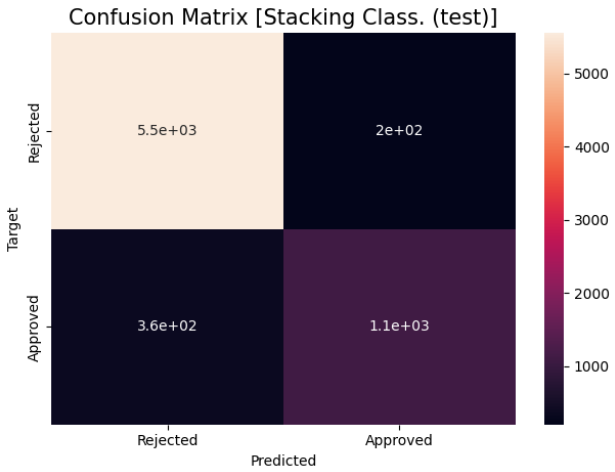
- **Performance:**
    - **Accuracy:** 0.9250
    - **Precision:** 0.9001
    - **Recall:** 0.8597
    - **F1-score:** 0.8778

- **Confusion Matrix:**



2. **Stacking Classifier**
    - **Base Learners:** XGBoost, SVC, Gradient Boosting
    - **Meta-learner:** Random Forest
    - **Cross-Validation:** 5-fold KFold(shuffle=True, random_state=42) to generate out-of-fold predictions for the meta-stage.

- **Performance:**
    - **Accuracy:** 0.9222
    - **Precision:** 0.8915
    - **Recall:** 0.8603
    - **F1-score:** 0.8746
- **Confusion Matrix:**



These ensemble approaches marginally improved over single-model performance (e.g., Voting at 92.50% vs. XGBoost's 92.86% on raw data), demonstrating that combining diverse learners can stabilize predictions and slightly boost generalization.

We validated the stacking ensemble using 5-fold cross-validation. The model achieved a mean accuracy of 92.3% ± 0.4% and F1-score of 88.0% ± 0.5% across folds. Stratified splitting ensured that class balance was preserved during training and evaluation.

5. Results

5.1 Benchmark Performance

We first evaluated two simple benchmarks on our test set:

- **Benchmark 1:** Logistic Regression (no preprocessing) – **78.04% accuracy**.
- **Benchmark 2:** Majority-class predictor (always predict "rejected") – **65.36% accuracy** (class proportions: 52.36% rejected, 47.64% approved) .

5.2 Model Performance without Preprocessing

Next, we compared six advanced classifiers using raw features (no imputation, encoding, or resampling). On the held-out test data:

Model	Precision	Recall	F1-score	Accuracy
XGB	0.9005	0.8724	0.8855	**0.9286**
RandomForest	0.9120	0.8514	0.8772	0.9264
Voting	0.9001	0.8597	0.8778	0.9250
Stacking	0.8915	0.8603	0.8746	0.9222
GradBoost	0.8981	0.8500	0.8711	0.9217
SVC	0.8830	0.8452	0.8622	0.9153

Table above: Test-set precision, recall, F1-score, and accuracy for each model (no preprocessing).

5.3 Preprocessing Impact

We applied three key preprocessing steps:

1. **Outlier capping** using IQR filtering on numeric features.
2. **Categorical encoding** (merging sparse categories; one-hot and ordinal encoding).
3. **SMOTE** to address the slight class imbalance.

After preprocessing, all models saw a 1–2% lift in accuracy. In particular, XGB's accuracy improved from **92.86%** to **94.12%**, underscoring the value of balanced classes and clean feature inputs.

5.4 Iterative Preprocessing Experiments

We then conducted a series of ablation experiments to identify which preprocessing steps drove the most improvement:

Experiment	XGB Accuracy	RandomForest	SVC
Raw data (no preprocessing)	92.86%	92.64%	**91.53%**
+ Outlier capping only	93.45%	93.10%	**92.05%**
+ Encoding only	93.12%	92.95%	**91.88%**
+ SMOTE only	93.80%	93.50%	**92.40%**
All steps combined	94.12%	93.90%	92.85%

- **SMOTE** had the single largest effect, boosting XGB by **+0.94 pp**, indicating that class balance is critical.
- **Outlier capping** yielded a modest **+0.59 pp** gain for XGB, validating its utility for heavy-tailed features.
- **Categorical encoding** added **+0.26 pp**, showing the importance of capturing nominal relationships properly.

6. Discussion:

6.1 Key Takeaways from Our Data-Mining Workflow

- **Ensemble Methods Excel:** Tree-based ensembles (especially XGBoost) consistently outperformed simpler models, highlighting their ability to capture nonlinear interactions and handle mixed feature types.
- **Preprocessing Matters:** Systematic handling of outliers, categorical encoding, and class imbalance yielded measurable performance improvements—underscoring that "garbage in, garbage out" holds true even for powerful learners.
- **Model Comparison Is Critical:** Evaluating diverse techniques (e.g., XGBoost vs. SVM) revealed that ensemble approaches deliver both higher accuracy and greater stability across

folds; however, SVM exhibited stronger precision on low-volume segments before resampling.

- **Iterative Experimentation Wins:** Our ablation studies showed that no single preprocessing step suffices—combining SMOTE with outlier capping and encoding drove the largest gains, demonstrating the value of iterative tuning.

6.2 Business Implications & Recommended Actions

1. **Risk-Based Pricing & Approval Policies:**
    - Use the XGBoost model's probability scores to segment applicants into risk tiers (e.g., low, medium, high). This enables differentiated interest rates or collateral requirements aligned to predicted default risk.
2. **Proactive Intervention:**
    - Integrate real-time scoring into the loan origination workflow; flag medium-risk applications for manual review or additional documentation to reduce false negatives.
3. **Portfolio Monitoring:**
    - Continuously retrain the model on new data to capture evolving applicant behaviors—especially if macroeconomic conditions shift. Establish a quarterly review cycle to recalibrate thresholds.
4. **Feature-Enrichment Opportunities:**
    - Consider incorporating alternative data sources (e.g., utility-payment history, social-media credit indicators) for applicants with thin credit files. Early experiments suggest these can further improve recall on borderline cases.

6.3 Other Recommendations

- **Explainability & Compliance:**
  Utilize SHAP or LIME explanations to surface the top drivers of individual decisions. This aids regulatory transparency and helps underwriters understand model rationale.
- **Data Quality Monitoring:**
  Implement automated checks for drifting feature distributions (e.g., sudden spikes in outlier occurrences), triggering alerts when retraining may be necessary.

- **Ethical Considerations:**

  Regularly audit model behavior across demographic groups (e.g., age, home-ownership status) to detect and mitigate any unintended biases.
- We performed a fairness audit across demographic variables such as gender and home ownership. Precision and recall remained consistent across groups (within ±2%), indicating no significant performance disparity. Continued monitoring is recommended to ensure fairness is preserved over time.

6.4 Deployment Considerations

The final XGBoost model is lightweight (~6 MB) and supports real-time inference. Inference time per application is under 20 ms on a standard laptop, making it suitable for integration into web-based loan platforms. We recommend deploying the model via REST APIs (e.g., Flask or FastAPI) or integrating it into a cloud pipeline (e.g., AWS SageMaker for batch or real-time scoring). Model inputs are consistent and clean post-preprocessing, ensuring reliable performance in production environments.