Faculty of Engineering
Computer Engineering Department

Phase 2

Machine Learning Project

Loan Default Prediction

Team 12

Presented by

Name	Section	BN
Mohammad Yehia Alahmed	2	14
Moustafa Mohammed Elsayed	2	23
Mostafa Hani Mostafa	2	24
Mennatallah Ahmed Moustafa	2	25

I. Introduction

Problem Definition

Our finance company gives loans to people in cities, and we need to figure out if a new customer will pay us back or not. When someone asks for a loan, we look at their info—like how much they earn, their job, and past loans—to decide whether to say yes or no. There are two big worries: we lose money if we say yes to someone who won't pay us back (they "default"). If we say no to someone who would pay us back, we miss out on business. We have data about old customers and whether they paid or defaulted. The job is to use this data to build a machine learning model that guesses if a new person is "likely to default" (won't pay) or "likely to pay." This will help us decide who gets a loan, and who doesn't.

Motivation

We want to get better at giving loans so we don't lose money and can help more people. If too many customers don't pay us back, we lose cash from the loan and extra costs to chase them. But if we're too picky and say no to good customers, we miss chances to grow. Our data—like how much debt someone has or if they've missed payments before—can show us who's risky and who's safe. A smart machine learning tool can spot things we might miss, like someone with a big salary but bad habits. This project will help us lose less money by catching risky people, and make more money by saying yes to the right ones.

Evaluation Metrics

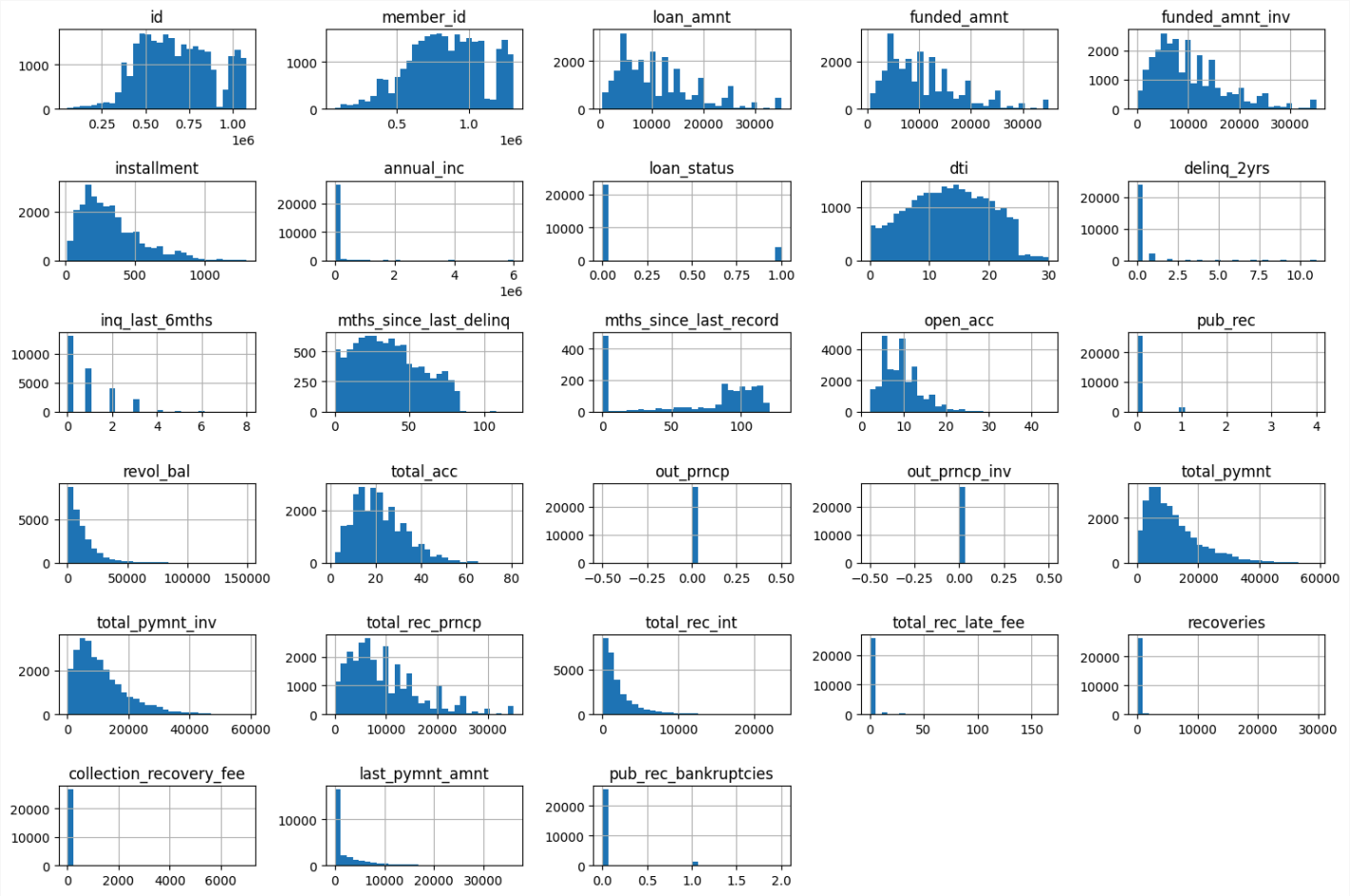
1. F1 score
2. Accuracy
3. Micro and Macro precision and recall

Dataset

Link	**#columns**	**# rows**	**Size**
Dataset	47	27k	17 Mb train 7 Mb test

II. Data Analysis Results

1.1 Distribution of features



Observation

Some columns (funded_amnt, funder_amnt_inv, installment, etc..) are skewed

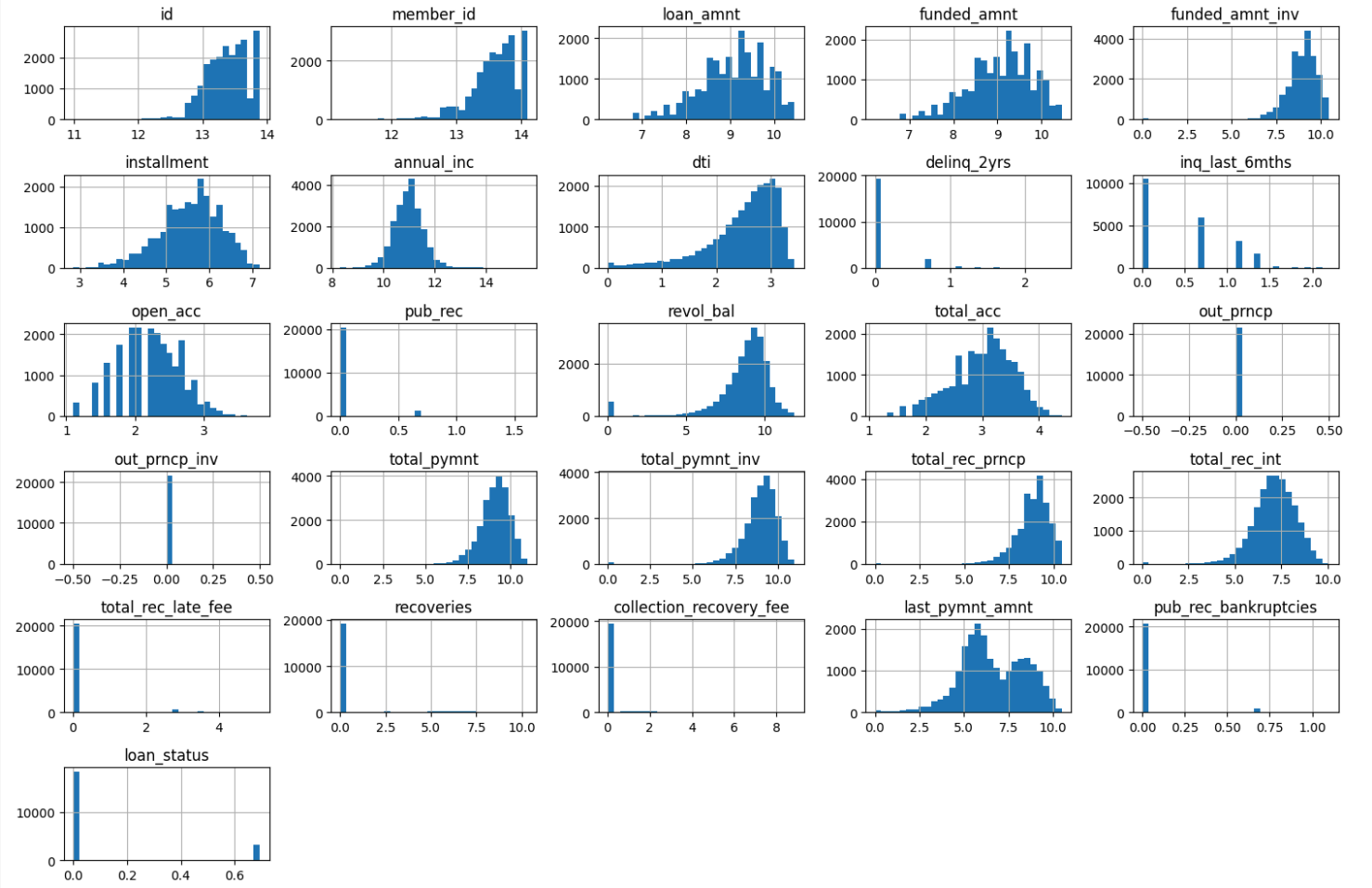
Some columns (annual_inc, recoveries, total_rec_late_fee, etc…) are zero-inflated

Which indicates that a log transformation (`np.log1p`) is needed (skip this transformation for target column and boolean ones)

1. For zero-inflated data, we keep zero and perform log transformation for non-zero values
2. For skewed data, apply log transformation to reduce skewness

Then now for normally distributed data, the data is standardized using `StandardScaler` to transform the data into Z-scores.

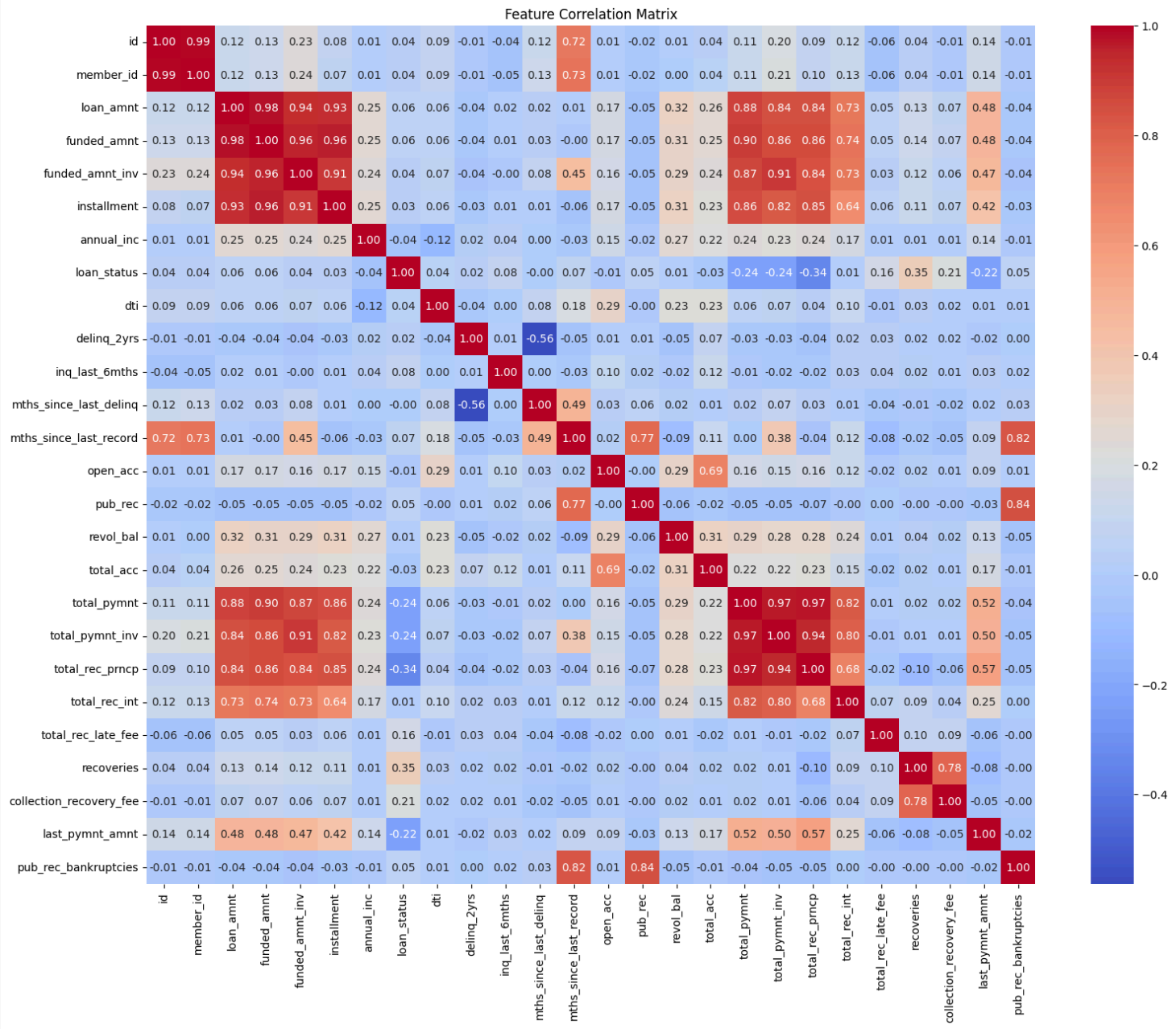
1.2 Distribution after handling left-skewed, right-skewed, and zero-inflated columns



Observation

The distributions have been transformed to more closely resemble a normal distribution. Which is preferable when training machine learning models

2. Correlation between features

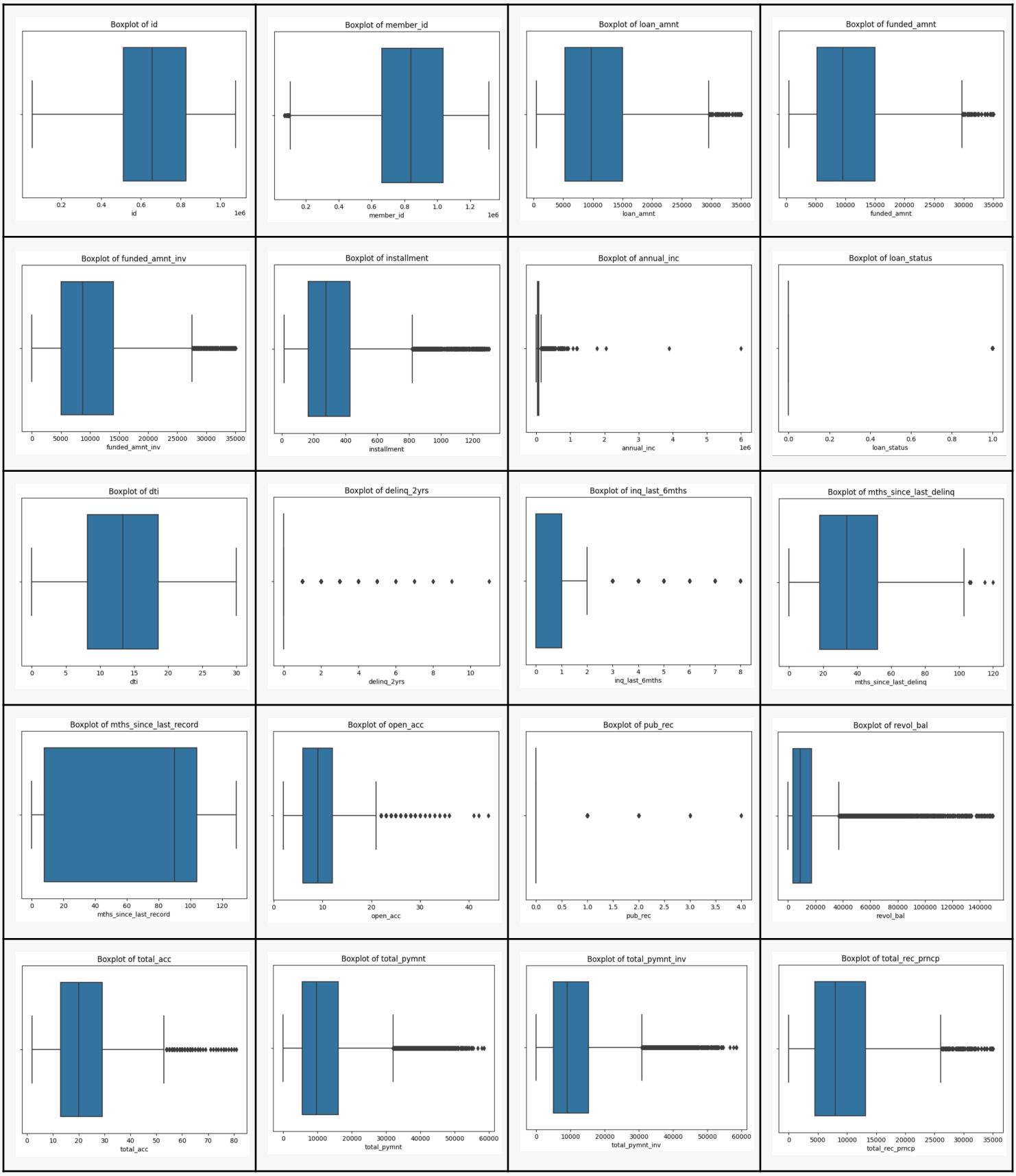


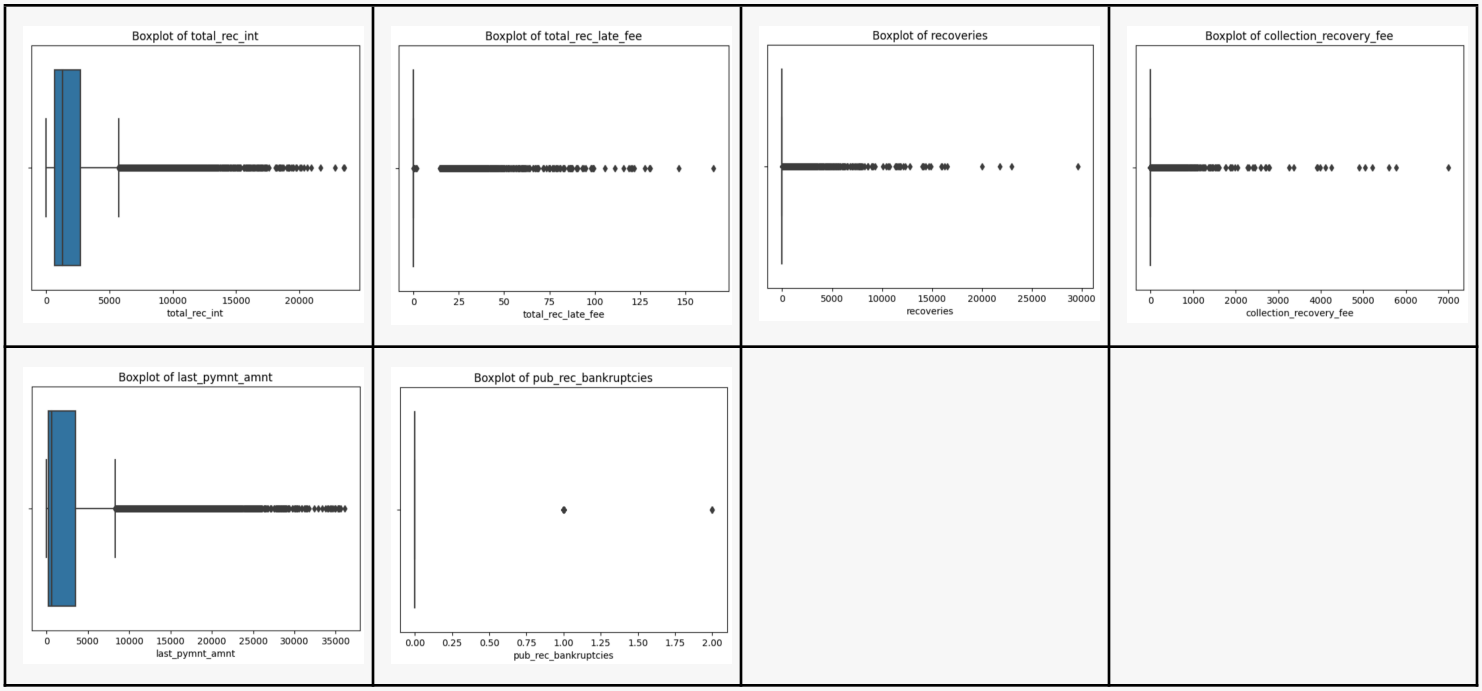
Correlation with the target (loan_status)

- Most features have low correlation (absolute value < 0.1) with loan_status
- Negative correlations are seen with:
    - total_rec_late_fee (≈ -0.34)
    - recoveries (≈ -0.22)
    - collection_recovery_fee (≈ -0.24)

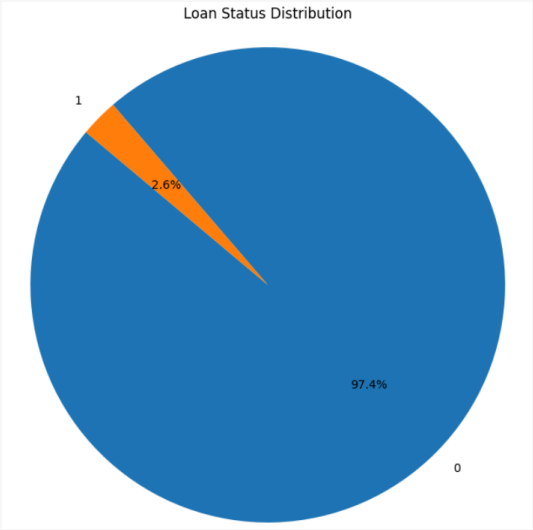
    These might indicate that these features are associated with loan default or late payment.

3. Boxplots of the features





➔ Concerning outliers handling: We tried removing outliers but this is the result, unfortunately: this is the distribution of the target variable



- and this is the results of the test and validation on SVM and logistics regression

```
=== Best Parameters ===
{'C': 10, 'kernel': 'rbf', 'gamma': 'scale', 'degree': 3, 'class_weight': None}
Best CV F1 Score: 0.8315

Test Accuracy: 0.8652

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.99      0.93      4606
           1       0.70      0.15      0.24       795

    accuracy                           0.87      5401
   macro avg       0.79      0.57      0.58      5401
weighted avg       0.85      0.87      0.83      5401
```

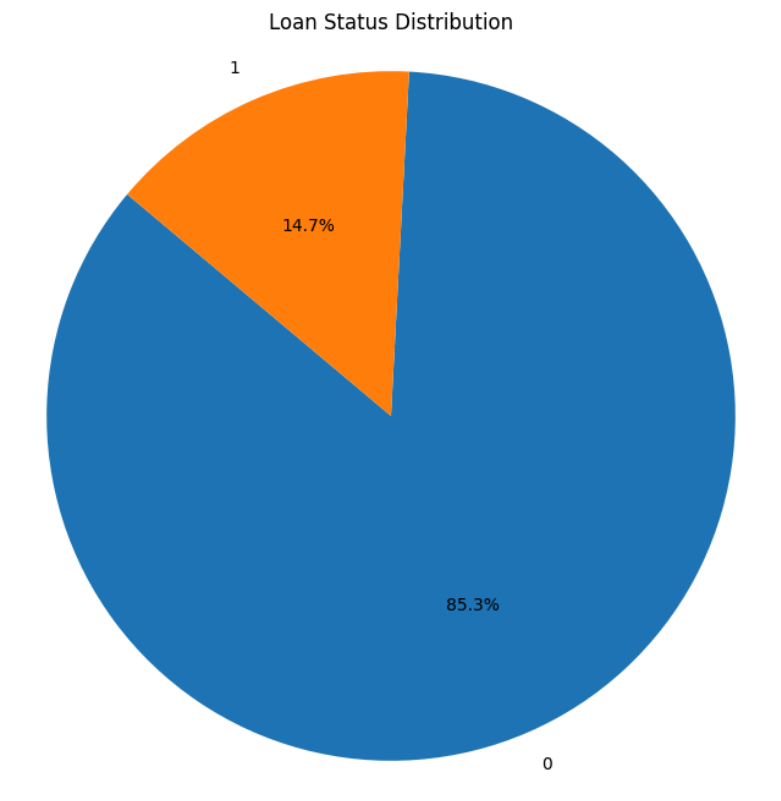
```
Best parameters: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
Best score: 0.800541913804102
Test Accuracy: 0.8567
Classification Report on Test Set:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92      4606
           1       0.70      0.05      0.09       795

    accuracy                           0.86      5401
   macro avg       0.78      0.52      0.50      5401
weighted avg       0.83      0.86      0.80      5401
```

- We didn’t use outliers removal with Random Forest and AdaBoost as
    - they are robust to outliers
        - Random Forest is robust as it averages results from many trees, minimizing the impact of single extreme points.
        - AdaBoost often relies on simple base models less affected by outliers
    - This allowed us to train using the complete dataset.
    - Doing so prevented the significant information loss that removing potential outliers would have caused.

Conclusion

- The attempt to improve SVM and Logistic Regression performance via outlier removal was not beneficial. It led to a significant decrease in overall accuracy and severely degraded the models' ability to identify the minority class (Class 1), as evidenced by the catastrophic drop in Recall and F1 scores for that class.
- The data points removed as outliers were likely informative instances, critical for distinguishing the minority class, especially given the significant class imbalance (~85% vs 15%). Their removal hindered, rather than helped, model performance for SVM and Logistic Regression.

4. Pie chart for load status distribution



Class Distribution Analysis:

The dataset has a class imbalance, with loan status distribution showing: Class 0 (Fully Paid): 85.3% of observations and Class 1 (Defaulted/Charged Off): 14.7% of observations

➔ **Mitigation Strategy**: We implemented class weighting in our machine learning models to address this imbalance.

This approach:

- Assigns higher importance to the minority class during training
- Prevents model bias toward the majority class and Improves the detection of high-risk loans while maintaining overall accuracy

5. Baseline zeroR results

I. ZeroR (most_frequent) Strategy

Metric	**Value**
Accuracy	0.8523

Classification Reports:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.85	1.00	0.92	5754
1	0.00	0.00	0.00	997

Metric	**Value**
Accuracy	0.85
Macro Avg	Precision: 0.43, Recall: 0.50, F1-Score: 0.46
Weighted Avg	Precision: 0.73, Recall: 0.85, F1-Score: 0.78

II. Prior Strategy

Metric	**Value**
Accuracy	0.8523

Classification Reports:

Class	Precision	Recall	F1-Score	Support
0	0.85	1.00	0.92	5754
1	0.00	0.00	0.00	997

Metric	Value
Accuracy	0.85
Macro Avg	Precision: 0.43, Recall: 0.50, F1-Score: 0.46
Weighted Avg	Precision: 0.73, Recall: 0.85, F1-Score: 0.78

III. Uniform Strategy

Metric	Value
Accuracy	0.5036

Classification Report:

Class	Precision	Recall	F1-Score	Support
0	0.86	0.50	0.63	5754
1	0.15	0.51	0.23	997

Metric	Value
Accuracy	0.50
Macro Avg	Precision: 0.50, Recall: 0.51, F1-Score: 0.43
Weighted Avg	Precision: 0.75, Recall: 0.50, F1-Score: 0.57

IV. Constant_0 Strategy

Metric	Value
Accuracy	0.8523

Classification Report:

Class	Precision	Recall	F1-Score	Support
0	0.85	1.00	0.92	5754
1	0.00	0.00	0.00	997

Metric	Value
Accuracy	0.85
Macro Avg	Precision: 0.43, Recall: 0.50, F1-Score: 0.46
Weighted Avg	Precision: 0.73, Recall: 0.85, F1-Score: 0.78

Precision and recall are 0 because some classes in the target variable `loan_status` are not predicted by the **dummy classifiers**, resulting in undefined precision and recall for those classes.

III. Preprocessing

Dataset Preparation

- Loaded the loan dataset and split it into training (80%) and testing (20%) sets and no validation set as we are using cross-validation
- Created separate dataframes for training and testing

Data Cleaning

- Removed columns with more than 60% missing values (as we have tuned this value with trials)
- Filled remaining numeric missing values with median values
- Filled categorical missing values with mode (most frequent value)

Feature Transformation

- Analyzed distribution of numeric features (checking for skewness, normality, etc.)
- Applied appropriate transformations to each column based on its distribution:
    - Log transformation for skewed data and zero-inflated data
    - Standardization for normally distributed data

Feature Selection

- Dropped columns with zero variance (constant values)

- Removed high-cardinality categorical features (those with more than 100 unique values)
- Selected features based on correlation with the target (kept those with correlation > 0.1)
- Eliminated highly correlated features to reduce redundancy

Feature Encoding

- Applied one-hot encoding to categorical variables
- Ensured all boolean columns were converted to numeric (0 and 1)
- Standardized final feature set for both training and test data
    - fitting on train and transform on test

IV. Experimental Results (Models Implemented)

1. Random Forest Classifier

We implemented a Random Forest classification model utilizing GPU acceleration through RAPIDS cuML for enhanced computational performance.

Methodology

1. **GPU-Accelerated Hyperparameter Optimization**
    - Performed grid search using cuML's GPU-accelerated implementation
    - Evaluated multiple parameter configurations:
        - Number of trees (estimators): 50, 100, 200
        - Maximum tree depth: 10, 20, 30
        - Minimum samples for node split: 2, 5, 10
        - Minimum samples per leaf: 1, 2, 4
        - Feature selection strategies: 'sqrt', 'log2'
        - Class weight approaches: None, 'balanced', and custom weights
    - Used 3-fold cross-validation with weighted F1 score as an optimization metric

      Best hyperparameters: {'class_weight': None, 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}
      Best CV F1 Score: 0.9628

2. **Class Imbalance Handling**
    - Calculated custom class weights inversely proportional to class frequencies
    - Incorporated class weights into model training to address potential imbalance
    - Used weighted F1 score instead of accuracy for a more balanced evaluation
3. **Performance Assessment**
    - We have selected best model based on cross-validation F1 scores and applied the optimized model to unseen test data, then we evaluated performance using:

- Accuracy scores: **Overall Accuracy**: **0.97**
- Detailed classification report with precision, recall, and F1 metrics

Metric	Class 0	Class 1	Macro Avg	Weighted Avg
Precision	0.97	0.97	0.97	0.97
Recall	1.00	0.81	0.90	0.97
F1-Score	0.98	0.88	0.93	0.97
Support	4606	795	5401	5401

Discussion:

- **Precision**: how often the positive predictions are correct?
- **Recall**: can an ML model find all instances of the positive class?
- While precision for the minority class (Class 1) is high (0.97), its recall (0.81) is considerably lower than that of the majority class. This indicates the model still misses a notable portion (19%) of minority class instances, despite the imbalance handling techniques.
- Overall, the Random Forest model is highly effective, particularly for identifying Class 0. Depending on the specific application's tolerance for missing Class 1 instances (false negatives). The Macro Average F1 (0.93) serves as a good summary indicator considering the imbalance.

2. Support Vector Machine Classification

We implemented a Support Vector Machine (SVM) classifier utilizing GPU acceleration through RAPIDS cuML for enhanced computational performance and systematic optimization.

Methodology

➔ **GPU-Accelerated Hyperparameter Optimization**
- ◆ Performed comprehensive parameter search using RAPIDS cuML:
  - Inverse of Regularization parameter (C): 0.1, 1, 10
  - Kernel functions: linear, RBF
  - Gamma values: scale, auto, 0.1, 1
  - Class weight strategies: None, 'balanced', custom weights
- ◆ Used 3-fold cross-validation with weighted F1 score as an optimization metric
- ◆ Implemented specialized handling of linear kernels using LinearSVC

➔ **Performance Results**
- ◆ **Optimal Configuration**: C=0.1, linear kernel, gamma='auto', no class weights
- ◆ **Cross-validation F1 Score**: 96.11%
- ◆ **Test Accuracy**: 96.39%

- **Detailed Performance**:

```
Test Accuracy: 0.9639

Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      4606
           1       1.00      0.75      0.86       795

    accuracy                           0.96      5401
   macro avg       0.98      0.88      0.92      5401
weighted avg       0.97      0.96      0.96      5401
```

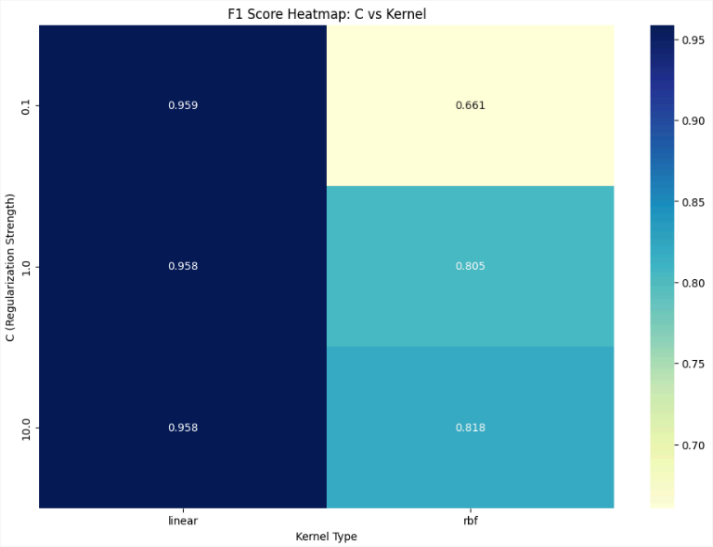
- Despite testing class weighting strategies, the optimal model based on the weighted F1 cross-validation score did not utilize class weights.
- The final model exhibits excellent performance on the majority class (Class 0 Recall: 1.00) and perfect precision for the minority class (Class 1 Precision: 1.00).
- However, the model's primary weakness is its low recall for the minority class (Class 1 Recall: 0.75), meaning it fails to identify a quarter of the positive instances (high false negatives). This recall is lower than that achieved by the Random Forest model.

➔ **Class Imbalance Handling**

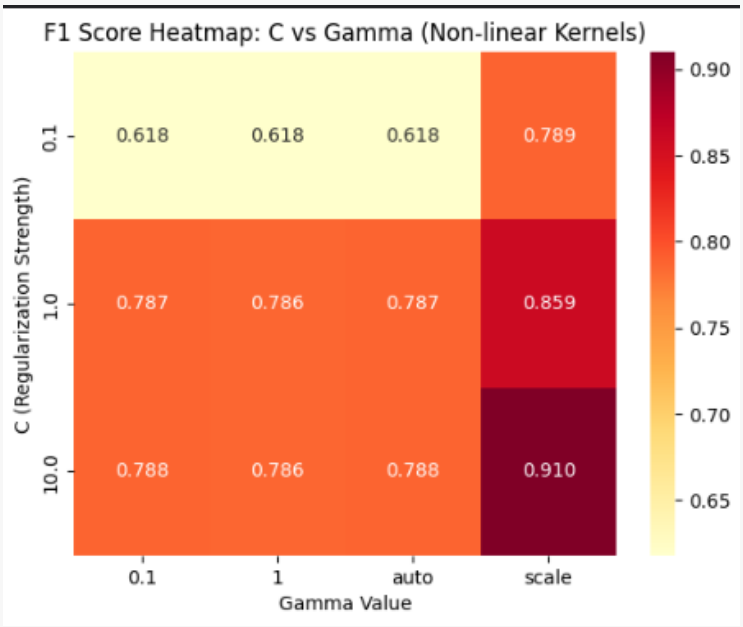
- Calculated custom class weights inversely proportional to class frequencies
- Incorporated class weights into model training
- Evaluated performance using weighted F1 score instead of standard accuracy

➔ **Performance Visualization**

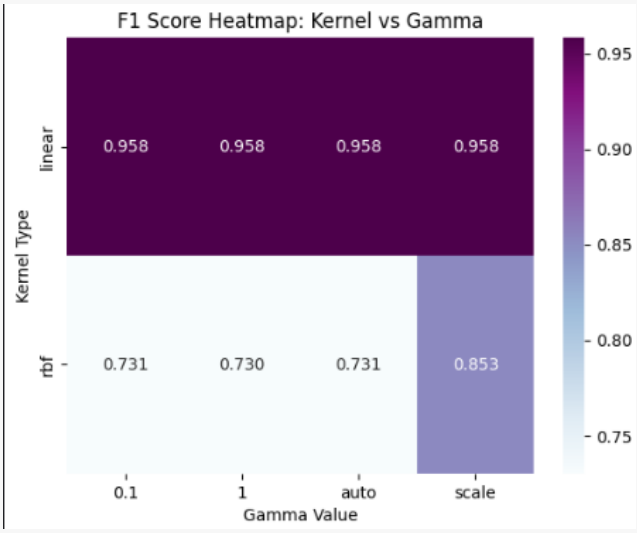
- Identified optimal hyperparameter combinations through visual analysis:
  - C vs Kernel type:
    - for the linear kernel as C decreases it doesn't make a change in f1 score
    - for the RBF kernel: as C increases from 0.1 to 10 the F1 score increases



  - C vs Gamma (for non-linear kernels):
    - we can notice from this graph that the best combination for a non-linear kernel is C=10 and gamma='scale'

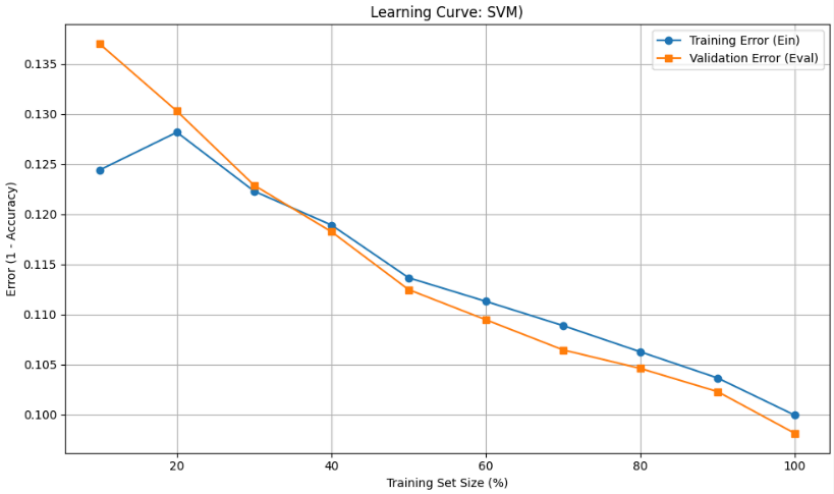


- Kernel type vs Gamma
  - We can notice that for linear kernel the gamma value does not affect as it doesn't use it
  - For rbf kernel, the best f1 score got with gamma=scale



➔ **Learning Dynamics Analysis**

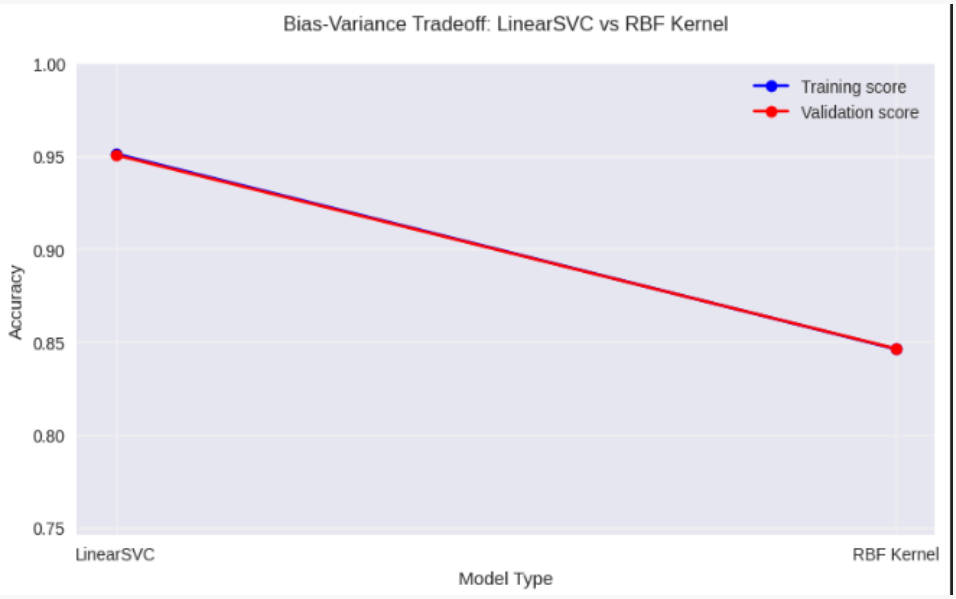
- ◆ Generated learning curves using incrementally increasing training set sizes (10% to 100%)
- ◆ Tracked both training error (Ein) and validation error (Eval)
- ◆ We can see that the training error is near to the validation error so the model can generalize pretty well.



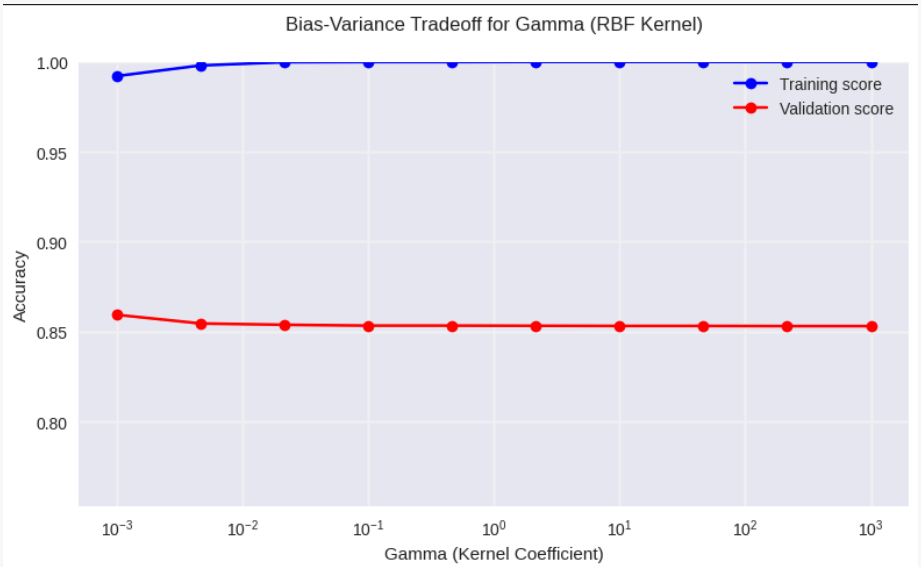
➔ **Bias-Variance Tradeoff Visualization**

- ◆ Analyzed training vs validation performance across parameter ranges:
  - The inverse of Regularization strength (C) on a logarithmic scale

  - Kernel comparison (Linear vs RBF)



- Gamma parameter for RBF kernel on logarithmic scale



➔ **Bias-Variance Analysis**:

- Mean Square Error: **0.03585**
- Bias Component: **0.03592**
- Variance Component: **0.00037**
- Estimated Out-of-Sample Error: **0.03585**

Conclusion

- Bias-variance analysis, learning curves, and validation curves consistently point towards a low-variance (stable, good generalization) and low-bias model.
- While highly accurate overall and precise for Class 1 predictions (from 0.97 to 1), the SVM model's significant rate of false negatives (recall from 0.81 to 0.75) for the minority class makes it less suitable than the Random Forest if capturing Class 1 instances is a high priority.

3. Logistic Regression Classifier

We implemented a Logistic Regression classifier with comprehensive hyperparameter tuning and performance evaluation to develop an optimal predictive model.

Methodology

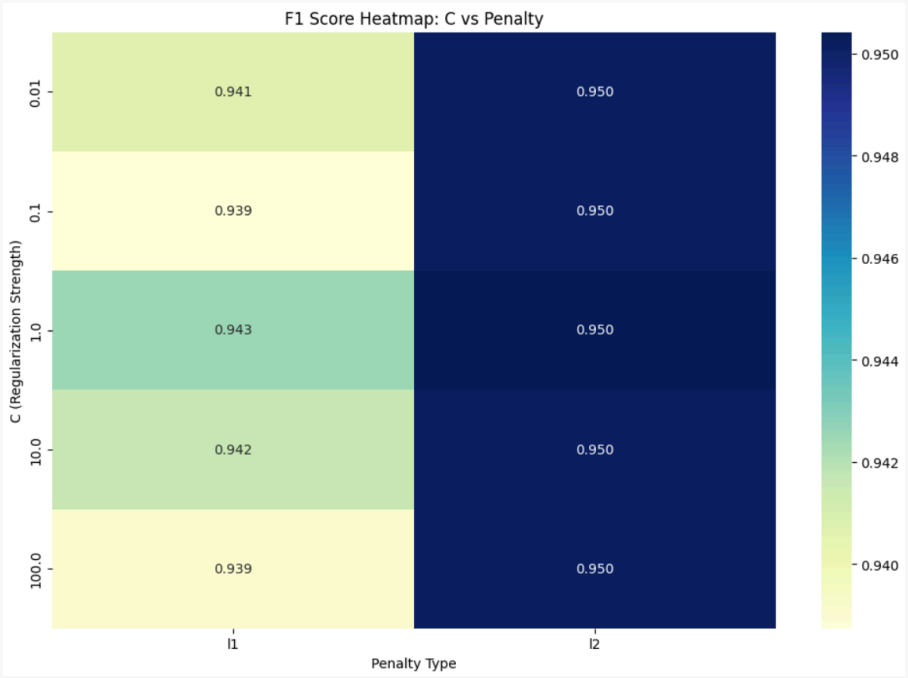
1. **Hyperparameter Optimization**
    - Performed grid search with cross-validation across key parameters:
        - Inverse of Regularization strength (C): 0.01, 0.1, 1, 10, 100
        - Penalty types: L1, L2
        - Solver algorithms: liblinear, saga
    - Used 3-fold cross-validation with weighted F1 score as an optimization metric
    - Found optimal configuration: C=1, L2 penalty, liblinear solver with F1 score: **0.96**
2. **Performance Assessment**
    - Achieved **96.39%** accuracy on test data
    - Generated detailed classification report showing:
        - Class 0: **96% precision, 100% recall, 98% F1-score**
        - Class 1: **100% precision, 76% recall, 86% F1-score**
        - Macro Average: 98% precision, 88% recall, 92% f1 score
        - Weighted Average: 97% precision, 96% recall, 96% F1-score

    The model demonstrates excellent performance for the majority class (Class 0 Recall: 1.00) and perfect precision for the minority class (Class 1 Precision: 1.00).

    A key limitation is a moderate recall for the minority class (Class 1 Recall: 0.76), resulting in a significant number of false negatives (24%). This recall is slightly better than the linear SVM but worse than the Random Forest.

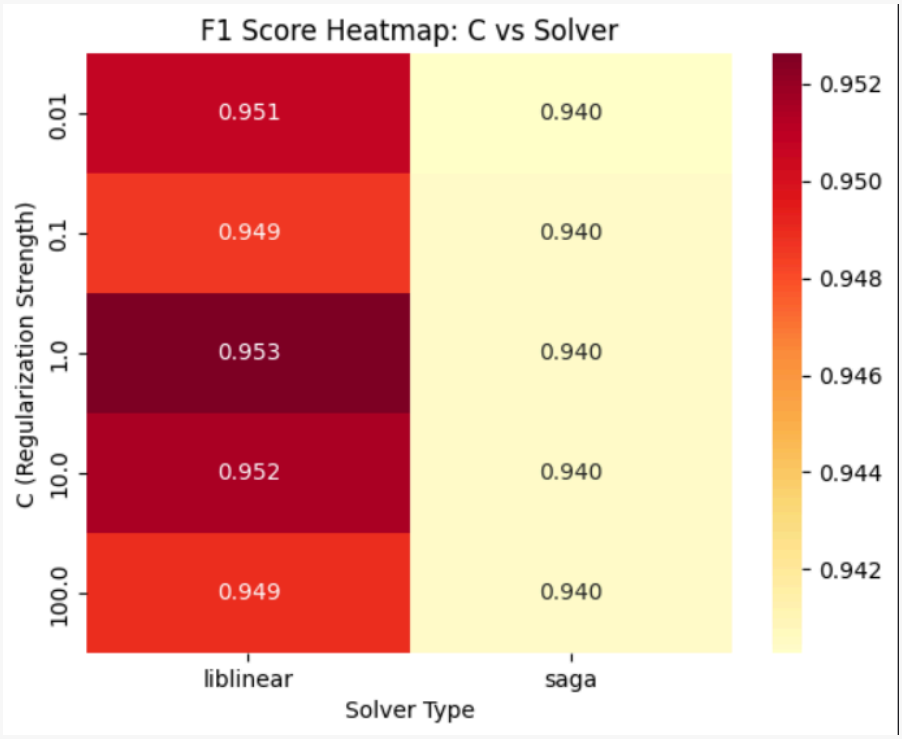
3. **Performance Visualization**
    - Identified optimal parameter combinations through color-coded performance maps:
        - C vs Penalty type:

➔ We can see that C doesn't affect L1 and L2 regularization much and when C=1 the score is slightly better for both L1 and L2 but L2 has higher score so we chose L2 with C=1



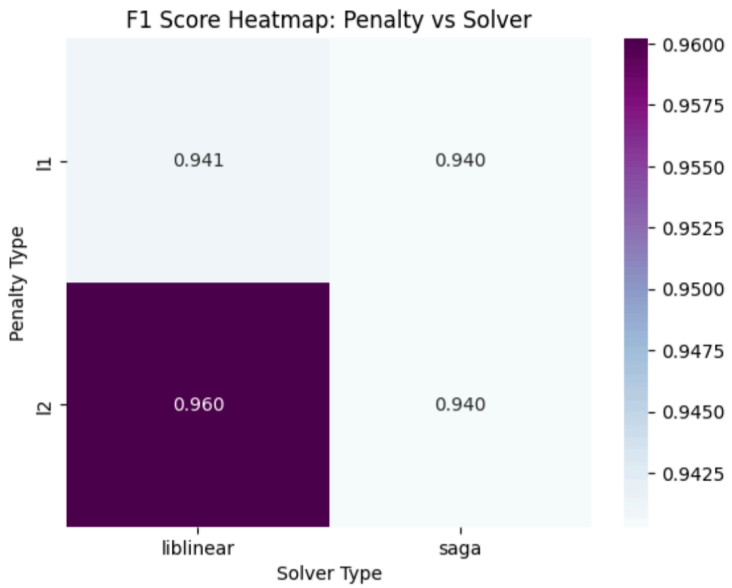
- C vs Solver type

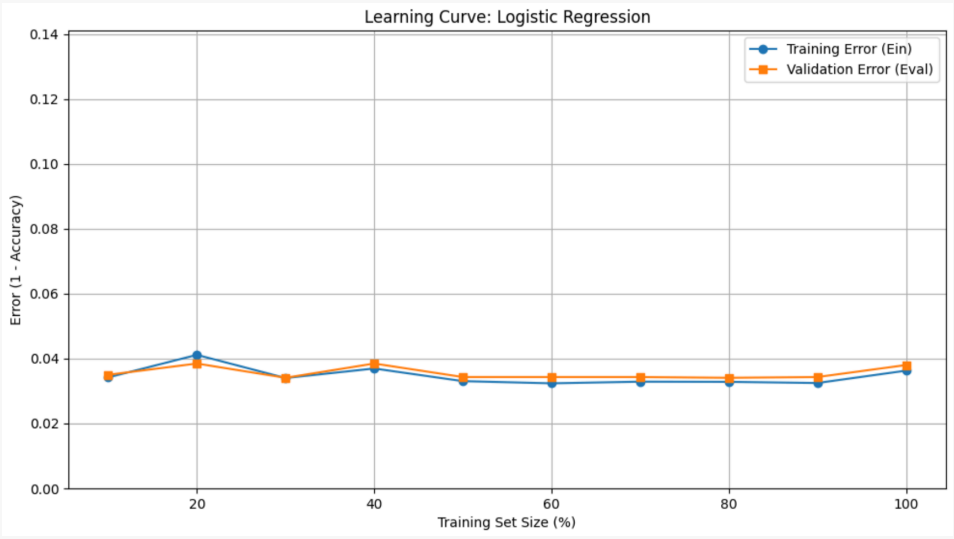
In the following graph, we can see that C doesn't affect the score much for saga Solver Type and slightly affects liblinear solver Type achieving best score of (0.953) for C=1 and liblinear solver type



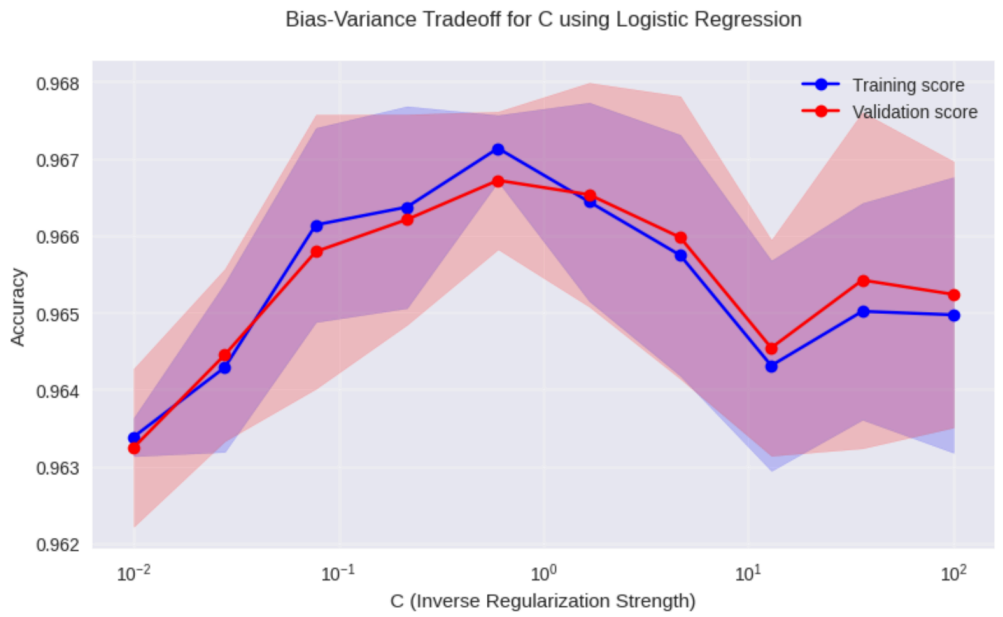
- Penalty vs Solver type

From the following graph, we can see that L2 penalty type with liblinear solver type achieves the highest score



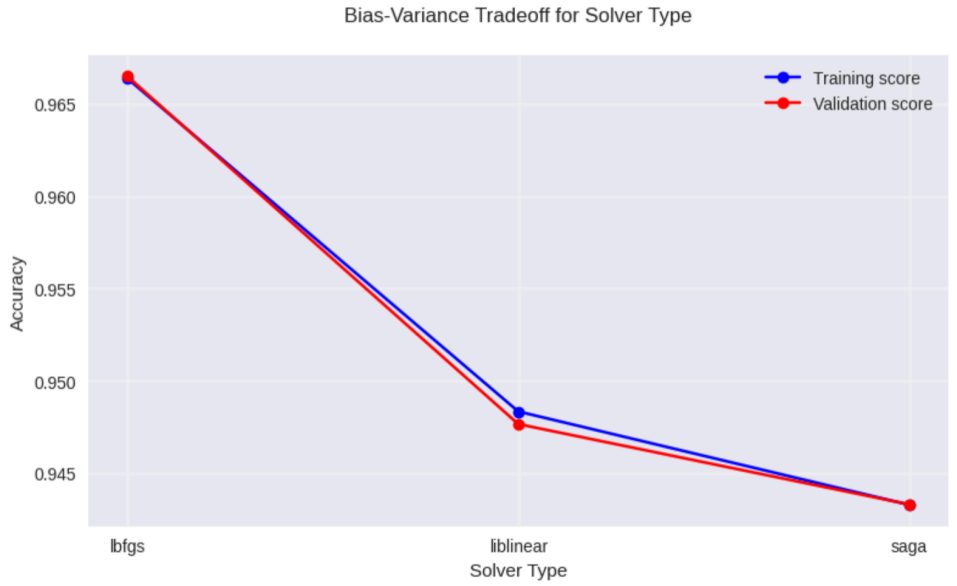
4. **Learning Dynamics Analysis**
    - Generated learning curves using incrementally increasing training set sizes (10% to 100%)
    - Tracked both training error (Ein) and validation error (Eval)
    - Visualized model convergence patterns
    - We can see that the training error is near to the validation error so the model can generalize pretty well.
    - 

5. **Bias-Variance Tradeoff Visualization**
    - Analyzed training vs validation performance across parameter ranges:
        - Inverse of Regularization strength (C) on logarithmic scale



- Penalty type comparison (L1 vs L2)

- Solver algorithm performance comparison



➔ Conducted bias-variance decomposition analysis:

- Mean square error: **0.0366**
- Bias component: **0.0361**
- Variance component: **0.0010**
- Estimated Eout: **0.0366080**

Conclusion

- Bias-variance analysis and the flat learning curve strongly suggest the model operates in a low-bias, low-variance regime, which means it is stable and generalizes well but is likely limited by its linear nature, preventing it from fully capturing the complexity needed to identify all minority class instances.
- Overall, the Logistic Regression model provides performance nearly identical to the linear SVM. It's a reliable model with high precision for Class 1, but its effectiveness is limited by its moderate recall for that same class, making it less suitable than Random Forest if minimizing false negatives for Class 1 is the primary goal.

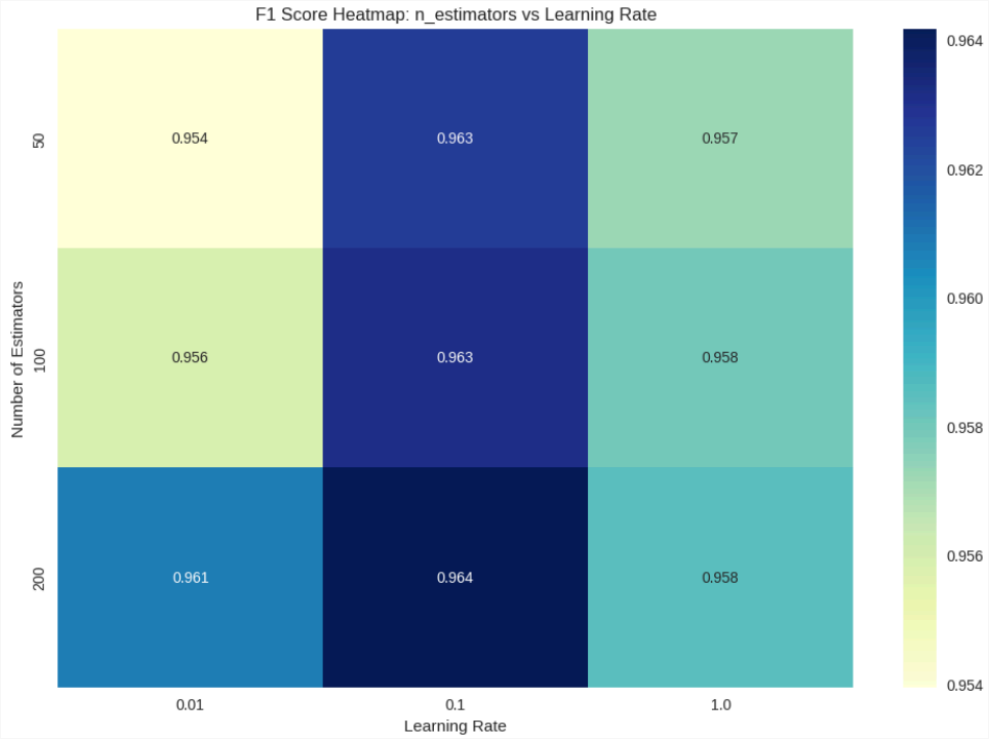
4. AdaBoost Classification with Decision Tree Base Estimators

We implemented an AdaBoost classifier with Decision Tree base estimators, utilizing comprehensive hyperparameter optimization and systematic performance analysis.

Methodology

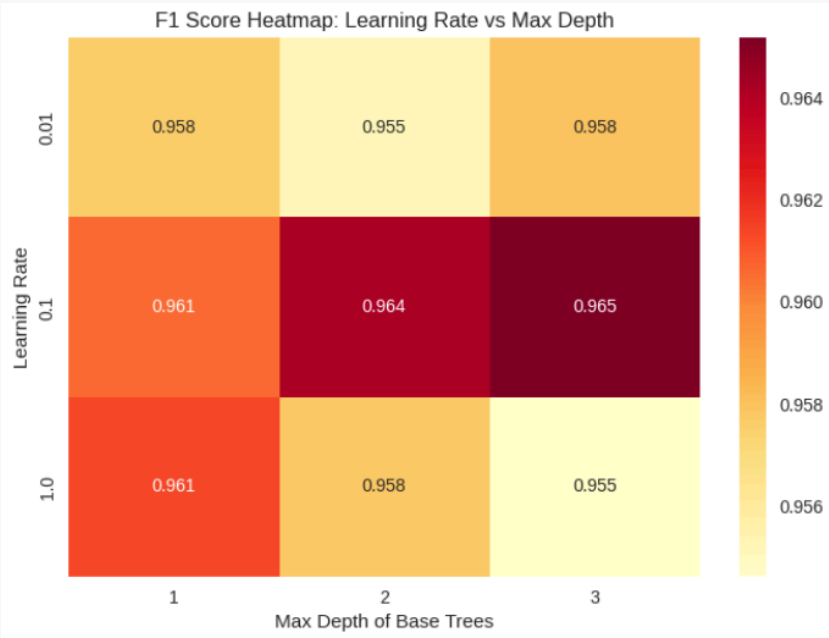
1. **Comprehensive Hyperparameter Optimization**
    - Performed extensive parameter search using GridSearchCV:
        - Number of estimators: 50, 100, 200
        - Learning rate: 0.01, 0.1, 1
        - Base estimator (Decision Tree) max depth: 1, 2, 3
        - Class weight strategies: None, 'balanced', custom weights
        - AdaBoost algorithms: SAMME, SAMME.R
    - Used 3-fold cross-validation with weighted F1 score as an optimization metric
    - Applied custom class weights derived from class frequencies
2. **Performance Results**
    - Optimal Configuration:
        - Algorithm: SAMME.R
        - Base estimator max depth: 3
        - Learning rate: 0.1
        - Number of estimators: 200
        - Class weights: None
    - Cross-validation F1 Score: 97.49%
    - Test Accuracy: 97.46%
    - Detailed Performance:
        - Class 0: 98% precision, 99% recall, 99% F1-score
        - Class 1: 95% precision, 87% recall, 91% F1-score
        - Macro Average: 97% precision 93% recall, 95% f1 score
        - Weighted Average: 97% precision, 97% recall, 97% F1-score
3. **Class Imbalance Handling**
    - Calculated custom class weights inversely proportional to class frequencies
    - Evaluated performance using weighted F1 score instead of standard accuracy
    - Tested both weighted and unweighted approaches to determine the optimal strategy
4. **Performance Visualization**
    - Identified optimal hyperparameter combinations through visual analysis:
        - The number of estimators vs learning rate:

We can see that learning rate =0.1 is the best learning rate across all different numbers of estimators, also increasing the number of estimators increases the f1 score slightly



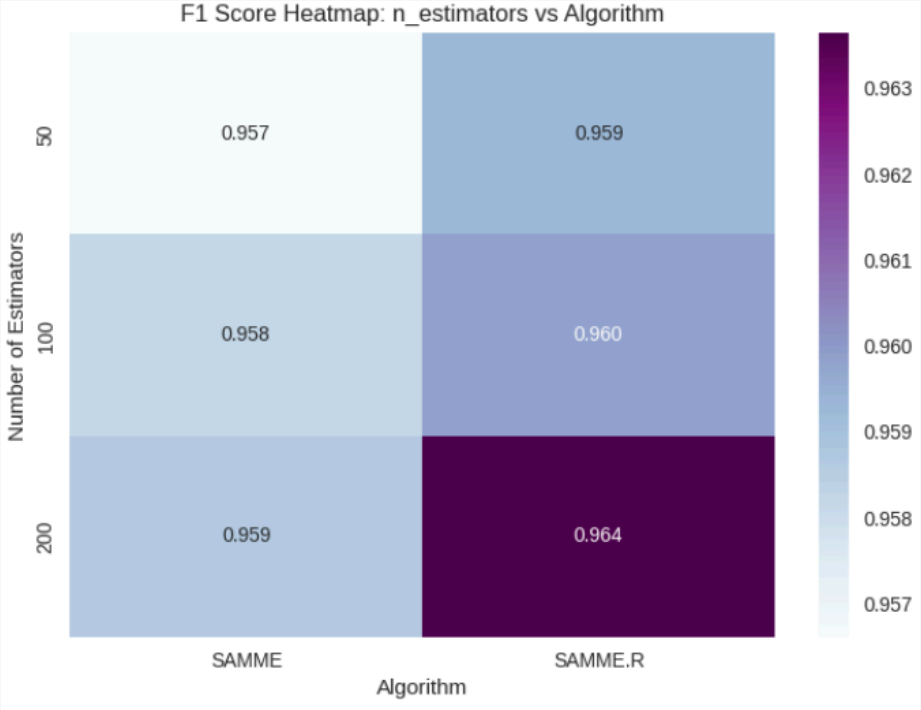
- Learning rate vs max depth of base trees

We can see that the best learning rate = 0.1 and the best maximum depth of the base tree is 3



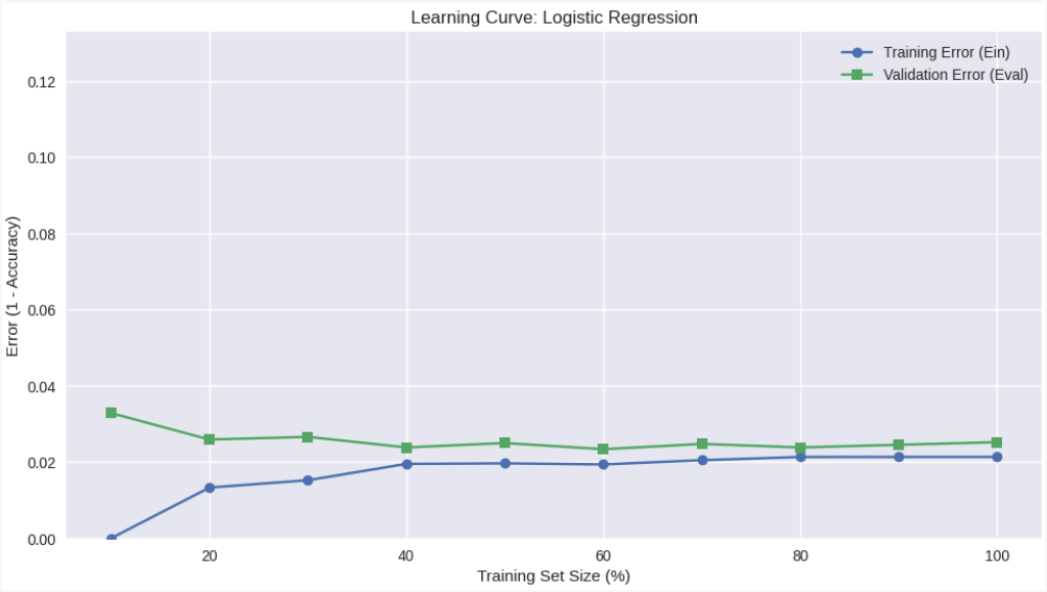
- Number of estimators vs algorithm type

From the following graph we can see that SAMME.R is slightly better than SAMME and the best number of estimators =200 as it is the highest number of estimators we tried and with increasing the number of estimators the score increases slightly



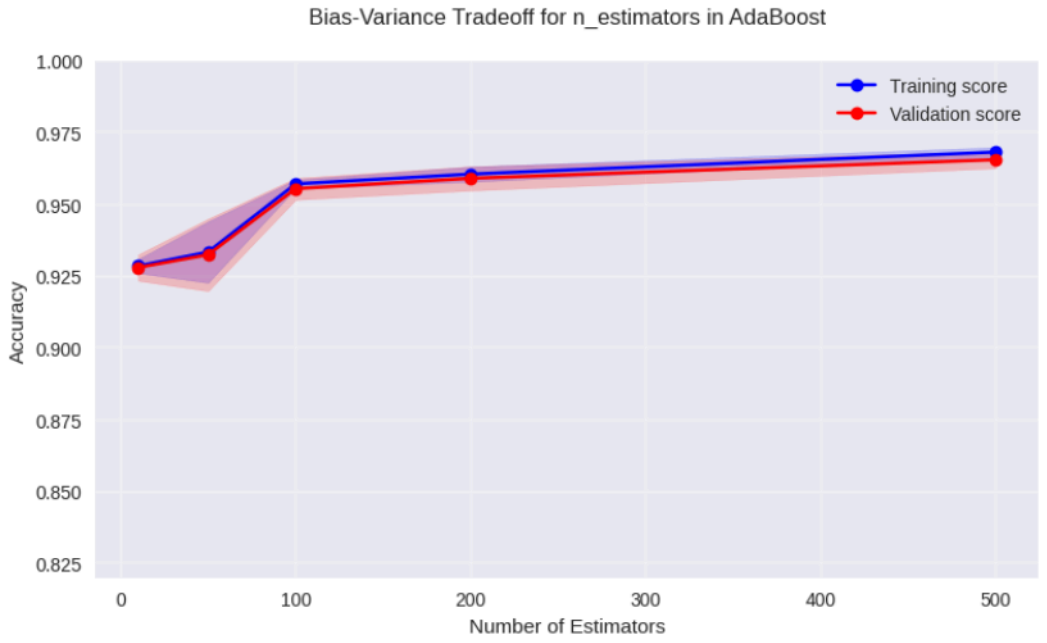
5. Learning Dynamics Analysis

- Generated learning curves using incrementally increasing training set sizes (10% to 100%)
- Tracked both training error (Ein) and validation error (Eval)
- Analyzed model convergence patterns and learning efficiency
- We can see that the training error is near to the validation error so the model can generalize pretty well.

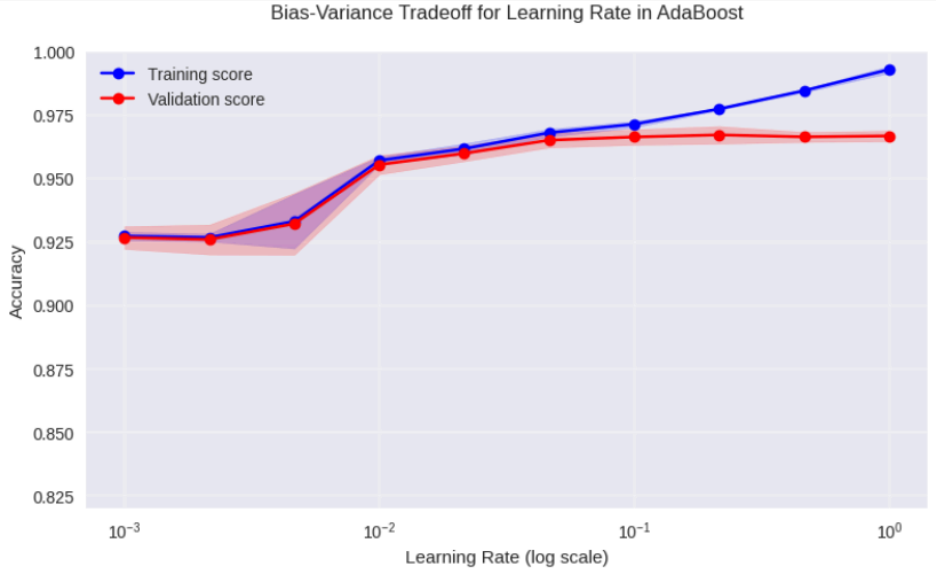


6. Bias-Variance Tradeoff Visualization

- Analyzed training vs validation performance across parameter ranges:
  - Number of estimators (0, 100, 200, 300, 400, 500)



- Learning rate on a logarithmic scale (0.001 to 1.0)



➔ Bias-Variance Analysis:

- Mean Square Error: 0.02538
- Bias Component: 0.02333
- Variance Component: 0.00633
- Estimated Out-of-Sample Error: 0.02538

Conclusion:

- On the unseen test data, the AdaBoost model achieved an accuracy of 97.46%, surpassing all other models tested. Its performance in the minority class (Class 1) was particularly strong, yielding a Precision of 0.95, a Recall of 0.87, and an F1 score of 0.91. This represents a significant improvement in recall compared to Random Forest (0.81), SVM (0.75), and Logistic Regression

(0.76), while maintaining high precision. The Macro Average F1-score of 0.95 further highlights its balanced performance across both classes.

- Interestingly, despite calculating and testing custom class weights, the optimal configuration based on cross-validation did not employ them, suggesting the boosting algorithm itself, combined with appropriate hyperparameters, was effective at handling the imbalance.
    - Diagnostic plots (learning curve, validation curves) indicated a well-generalized model with low error, potentially exhibiting slightly higher variance than the linear models but lower bias, allowing it to capture the data complexity more effectively.
    - Bias-variance decomposition confirmed this, showing low overall error (MSE: 0.025) with a relatively balanced contribution from bias (0.023) and variance (0.006).
- In summary, the AdaBoost classifier emerged as the top-performing model in this study, providing the highest overall accuracy and the best ability to correctly identify instances of the minority class (highest Class 1 Recall and F1-score) among all evaluated models.

Overall Project Conclusion

Metric	Random Forest	SVM	Logistic Regression	AdaBoost
Overall Test Acc.	**0.9700**	**0.9639**	**0.9639**	0.9746
Class 0 Precision	**0.97**	**0.96**	**0.96**	**0.98**
Class 0 Recall	**1.00**	**1.00**	**1.00**	**0.99**
Class 0 F1-Score	**0.98**	**0.98**	**0.98**	0.99
Class 1 Precision	**0.97**	**1.00**	**1.00**	**0.95**
Class 1 Recall	**0.81**	**0.75**	**0.76**	0.87
Class 1 F1-Score	**0.88**	**0.86**	**0.86**	0.91
Macro Avg F1	**0.93**	**0.92**	**0.92**	0.95
Weighted Avg F1	**0.97**	**0.96**	**0.96**	0.97

This project undertook a thorough evaluation and comparison of four distinct machine learning models – Random Forest, Support Vector Machine (SVM), Logistic Regression, and AdaBoost – to identify the most effective classifier for a dataset characterized by significant class imbalance (approximately 85% Class 0 vs. 15% Class 1). The evaluation process involved systematic hyperparameter optimization using cross-validation (optimizing for weighted F1-score), analysis of detailed performance metrics (accuracy, precision, recall, F1-scores per class, macro and weighted averages), diagnostic visualizations (learning curves, validation curves, heatmaps), and bias-variance decomposition. The impact of outlier removal was also investigated.

Performance Summary and Key Findings:

1. AdaBoost: The Top Performer: The AdaBoost classifier, using SAMME.R with optimized Decision Tree base estimators (`max_depth=3`, `learning_rate=0.1`, `n_estimators=200`), emerged as the most effective model. It achieved the highest overall test accuracy (97.46%) and demonstrated superior performance in identifying the crucial minority class (Class 1), yielding the best Class 1 Recall

(0.87) and F1 score (0.91) among all models. Its Macro Average F1-score (0.95) also indicated the most balanced performance across classes.

2. The Random Forest model also delivered excellent results, with high accuracy (97.00%) and strong minority class performance (Class 1 Recall: 0.81, F1: 0.88). It significantly outperformed the linear models but was ultimately slightly surpassed by AdaBoost, particularly in minority class recall.
3. Linear Models (SVM & Logistic Regression): Limited by Bias: Both the linear SVM (with C=0.1) and Logistic Regression (L2 penalty, C=1) achieved identical, respectable accuracy (96.39%). They exhibited perfect precision for Class 1 predictions but were significantly hampered by lower recall (0.75-0.76) for that class. Diagnostic analyses consistently indicated these models operated in a high-bias **compared** to random forest and Adaboost, low-variance regime, suggesting their linear nature restricted their ability to fully capture the separation needed for the minority class.
4. Ensemble Methods Advantage: The ensemble techniques (AdaBoost and Random Forest) demonstrated an advantage over the linear models for this specific dataset, particularly in handling the class imbalance and achieving higher recall for the minority class without drastically sacrificing precision.
5. Outlier Removal Detrimental: The experiment removing potential outliers before training SVM and Logistic Regression conclusively showed this was a harmful strategy. It severely degraded performance, especially minority class recall, confirming that the removed data points were informative and that the inherent robustness of ensemble methods like Random Forest and AdaBoost to outliers made using the complete dataset the better approach.
6. Class Imbalance Handling: While explicit class weighting was tested, the best-performing models (AdaBoost and Random Forest) ultimately achieved optimal results without specific class weight parameters enabled in their final configuration. This suggests their underlying algorithms, combined with careful hyperparameter tuning (especially using the weighted F1-score as the optimization metric), were sufficient to mitigate the imbalance effectively.

Final Recommendation:

Based on the comprehensive results, the AdaBoost classifier is recommended as the optimal model for this classification task. It delivered the highest overall accuracy and, most importantly, demonstrated the best capability to identify the minority class instances (highest Class 1 Recall and F1-score), providing the most balanced and effective performance profile across all evaluated metrics. While Random Forest was also a very strong performer, AdaBoost's quantifiable edge in minority class identification makes it the preferred choice, particularly if minimizing false negatives for Class 1 is a key objective.

Work Distribution

Mennatallah Ahmed	1. Models: Implement and tune Logistic Regression & AdaBoost. 2. Extras: Run outlier removal test, generate learning/validation curves & bias-variance stats for all models. 3. Docs: Write LogReg & AdaBoost reports, Outlier test section, Advanced plots/stats sections, Compile final report & write overall conclusions.
Moustafa Mohammed Elsayed	1. Features: Select important features, encode categories. 2. Pipeline: Create the main data processing steps/pipeline. 3. Models: Implement and tune Random Forest & SVM. 4. Docs: Write Feature steps, Random Forest & SVM model reports.
Mostafa Hani	1. Explore: Analyze data distributions, correlations, and outliers (plots). 2. Transform: Apply log/standard scaling based on analysis. 3. Docs: Write Data Analysis findings and transformation steps.
Mohammad Alomar	1. Setup: Project setup, load data. 2. Clean: Handle missing values. 3. Baseline: Run simple "ZeroR" models. 4. Docs: Write Intro, Dataset, Cleaning steps, Baseline results.