IS5126 Team 4 - Loan Default Prediction

CHOOK WIN YAN, LALITHA RAVI, SREELAKSHMI
{A0290547X, A0268254X, A0268357N}
{e1326000@u.nus.edu , e1101553@u.nus.edu , e1101656@u.nus.edu}
National University of Singapore, Singapore

I. PROBLEM STATEMENT

Within the loan approval process, a borrower who fails to fulfill their repayment obligations is considered a default. This leads to financial losses for the bank. To proactively manage this risk and enhance lending effectiveness, we propose developing a machine learning model. This model will leverage historical loan data and be optimized through experimentation with various machine-learning techniques. Its primary objective is to accurately predict the likelihood of loan defaults for individual borrowers. By identifying high-risk profiles, the model empowers banks to make informed lending decisions, ultimately promoting financial stability. Code can be found at: [1]

II. EXPLORATORY DATA ANALYSIS AND DATA PREPROCESSING

We begin by understanding the dataset and then move on to cleaning it using different techniques to fix errors and inconsistencies. After that, we perform Exploratory Data Analysis (EDA) and data preprocessing to get it ready for machine learning tasks.

A. Dataset Overview

The dataset, sourced from Kaggle through Lending Club, comprises 396,030 records featuring 15 string columns and 12 numerical columns. The class distribution indicates that **19.24%** of loans end in default, while **80.76%** is fully charged. This disparity highlights a significant imbalance in the data.
These features encapsulate details about the loan applicant at the point of application, crucial for the lender's decision-making process. Notably, several features exhibit null values, with percentages as follows: *revol_util* (0.07), *mort_acc*(9.54), *pub_rec_bankruptcies*(0.14), *emp_title* (5.79), *emp_length*(4.62), and *title*(0.44).

B. Data Cleaning And Pre-processing

We cleaned and processed our data based on the following sequence of steps:

1) We begin by standardizing columns that have multiple data types to a uniform data type to ensure consistency.
2) Records with null values are dropped from each column without altering data distribution.
3) Recognizing redundancy, we eliminate the "grade" column to prevent multicollinearity issues since all relevant information is available in the *'subgrade'*.
4) Upon inspection, we find many instances where *'emp_title'* is empty but annual income is available. Consequently, we introduce a new category, *'self-employed'*, to address this scenario. However, due to the excessive number of unique *'emp_title'* values (nearly 173,105), we opt to drop the column.
5) *'Employment length'* values, originally ranging from "< 1 year" to "10+ years," are standardized to a consistent format of 0 to 10 years as per the data requirement.
6) To simplify *'home_ownership'* categories, we merge infrequently occurring ones like *'ANY OTHER'* and *'NONE'* into a single category, *'OTHER'*
7) Given its redundancy and high uniqueness, we decided to discard the "loan application title" column, which duplicates information found in another feature named *'purpose'*.
8) After completing the data cleaning process, the dataset emerges with zero null values, ensuring enhanced consistency.

C. Train Test Split and Label Encoding

The dataset is split into training and testing sets, with a one-third allocation to the test set. This ratio is suitable given the dataset's substantial size of 400K entries. Crucially, the split preserves the original class distribution, maintaining consistent proportions of class instances in both subsets. Label encoding is performed such that, class 1 represents "charged off" or "default," while class 0 indicates "fully paid" status.

D. Exploratory Data Analysis

1) Numerical Columns: We explore the correlation among numerical features, identifying interest rate, loan term (in months), and debt-to-income ratio (DTI) as strongly negatively correlated with loan status. As these metrics increase, the likelihood of loan default rises accordingly.

For each numerical feature, we plot distributions and use box plots to summarize statistics. In the case of highly skewed distributions, we remove extreme outliers contributing to less than 1% of the data to enhance risk assessment for loan default prediction. The figures 1 and 2 illustrate the data before and after the removal of extreme outliers for the numerical

[1]Github code Link: Github Link

feature *'Annual Income'*. In loan default prediction, precise

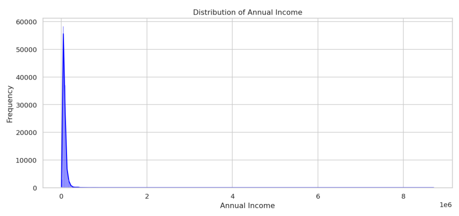

Fig. 1: Before extreme outlier removal for feature *'Annual Income'*

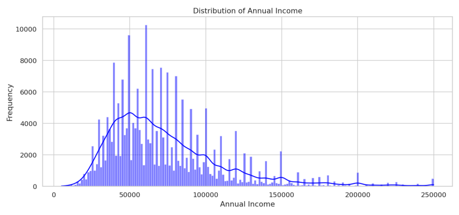

Fig. 2: After extreme outlier removal for feature *'Annual Income'*

risk assessment is vital for a lender's decision-making. By eliminating extreme outliers, lenders can focus on typical behavioral patterns, ensuring a more accurate evaluation of borrowers' creditworthiness.

2) Categorical Columns: We examine categorical variables to gauge the percentage of charged-off (default) records across different categories. This analysis aims to unveil predictive power within categorical features and discern underlying patterns essential for loan default prediction models. The Figure 3 shows the analysis done for the categorical feature *'sub-grade*

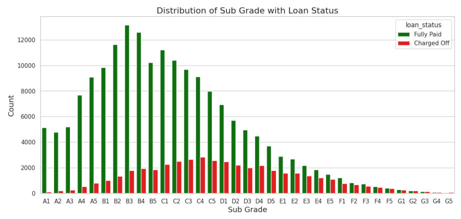

Fig. 3: Categorical analysis for feature *'sub-grade'*

3) Observations:

- The higher median for charge-off interest rates and installment amounts indicate increased chances of charge-off for higher values.
- Annual income exhibits significant skewness, leading to the removal of records with incomes exceeding \$250,000. Higher annual incomes generally correlate with a higher likelihood of full loan repayment.
- The *debt-to-income ratio (DTI)* is heavily skewed, prompting the removal of records with *DTI* exceeding 50. There's a noticeable increase in charge-off probabilities as *DTI* increases.
- The number of open accounts is highly skewed, resulting in the removal of records with more than 35 open accounts. However, the average open accounts do not vary significantly between charged-off and fully paid loans.
- An increase in the number of public records (*pub_rec*) correlates with higher chances of charge-off.
- In the case of open accounts, records with more than 2 accounts are removed due to skewness.
- The number of revolving balances (*revol_bal*) and revolving utilization (*revol_util*) are highly skewed, leading to the removal of records with balances exceeding \$150,000 and utilization exceeding 120.
- As *sub-grade* progresses it shows an increasing likelihood of charge-off.
- Charge-off percentages remain consistent across all employment length categories, indicating no significant predictive power causing drop-off.
- Variations in the percentage of charge-offs across different categories of other categorical columns suggest potential predictive power, warranting their retention in the analysis.
- For a detailed view of graphs and analysis refer to data_preprocessing in the code: Github Link

III. METHODOLOGY

A. Class Imbalance Handling

In our project, we address class imbalance to accurately predict loan defaults. We employ various data balancing techniques, including:

1) Random Undersampling:
   - We integrate undersampling into each fold of the hyperparameter tuning process using stratified k-fold cross-validation. This approach randomly removes data from the majority class to ensure a balanced representation with the minority class in every fold. This improves the model's generalizability and robustness and also prevents data leakage.
2) Model-Specific Techniques:
   - Logistic Regression: Used the *'class_weight'* parameter to penalize minority class misclassifications.
   - XGBoost: Utilized *'scale_pos_weight'* to assign higher weights to default cases.
   - Random Forest: Applied *'Balanced'* class weight balancing techniques to adjusts weights inversely to class frequencies.

B. Experimental Setup

This section details the experimental procedures employed to develop and optimize the machine learning model for predicting loan defaults. Throughout all experiments, we utilize **One Hot Encoding** to convert categorical variables into a binary format. The experiments can be broadly classified into 3 categories namely:

1) Experiment 1: Training on Entire Dataset:

*a) **Objective**:* This initial experiment establishes a baseline performance by training the model on the complete dataset without any adjustments to hyperparameters.

*b) **Process**:* The model is trained using all available data points. Default hyperparameter values provided by the chosen machine learning library are employed.

2) Experiment 2: Training on Entire Dataset with Hyperparameter Tuning:

*a) **Objective**:* This experiment aims to improve model performance by optimizing hyperparameters.

*b) **Process**:*

- Hyperparameter Tuning: We apply grid search to fine-tune simpler models such as logistic regression while employing randomized search techniques for more complex models like random forest and XGBoost due to increased complexity.
- Cross-Validation: Stratified K-fold cross-validation ensures robust evaluation by splitting the data into folds for training and testing.
- Scoring Metrics used to tune models: **F1-weighted** handles class imbalance by computing the harmonic mean of precision and recall for each class, and then averaging these values, weighted by the number of instances in each class. The**Average Precision Score** condenses a precision-recall curve by computing the weighted mean of precisions at various thresholds. It assigns weights based on the increase in recall from the previous threshold.

3) Experiment 3: Training on Entire Dataset with Hyperparameter Tuning and UnderSampling:

*a) **Objective**:* This experiment investigates the impact of under-sampling the majority class on model performance when dealing with class imbalance.

*b) **Process**:*

- Hyperparameter Tuning: Similar to Experiment 2, grid search and randomized search are used to optimize hyperparameters.
- Under-Sampling: During stratified K-fold cross-validation within each training fold, Random Undersampling removes instances from the majority class (likely non-defaults) to achieve a more balanced dataset.
- Scoring Metrics used to tune models: We use **F1-score** (without weighting due to balanced data from undersampling) and **Average Precision Score**.

C. Models Used

*1) **Naïve Bayes**:* Naïve Bayes was chosen as our baseline model due to its simplicity and assumption of feature independence, providing a straightforward starting point for performance evaluation.

a) Hyper-parameter Tuning: In the experimental setup, we used the Gaussian Naïve Bayes model for classification. Gaussian Naïve Bayes is chosen as it can effectively handle continuous data, by assuming that the input features follow a normal distribution. To optimize its performance, we performed hyper-parameter tuning using *RandomizedSearchCV* method. The goal was to find the best value that maximizes the model's performance. The parameter grid for tuning is as follows:

```
parameter_grid = {
'var_smoothing':[1e-9, 1e-8, 1e-7, 1e-6, 1e-5]
}
```

*2) **Random Forest**:* Random Forest, an ensemble learning technique, employs multiple decision trees to improve prediction accuracy and robustness compared to individual trees. By training each tree on random subsets of both the data and features and then combining their predictions through averaging, this method enhances predictive stability and accuracy.

a) Hyper-parameter Tuning: For optimization, the parameters we explored include n_estimators (tree count), max_depth (extensiveness of data split in each tree), min_samples_split (split threshold), min_samples_leaf (minimum leaf size), max_features (features per split) and bootstrap (sampling method). We used *RandomizedSearchCV* while configuring the class_weight parameter to "balanced". The parameter grid for tuning is given as:

```
param_grid = {
'n_estimators': [50, 100, 200, 400, 600],
'max_depth': [None, 5, 10, 15, 20, 30],
'min_samples_split': [2, 5, 10, 15, 20],
'min_samples_leaf': [1, 2, 4, 6, 8],
'max_features':['sqrt', 'log2', None, 0.5, 0.7],
'bootstrap': [True, False]
}
```

*3) **Logistic Regression**:* Given the relatively large dataset size, we opt for the Stochastic Gradient Descent (SGD) Classifier for logistic regression due to its efficiency in handling large-scale datasets. SGDClassifier updates the model parameters iteratively, making it suitable for datasets with a high number of samples and features. Since logistic regression is used, numerical values are scaled using various methods to ensure model stability and convergence.

We explore the below-mentioned scaling techniques for each ML experiment:

- RobustScaler: Scales features using statistics that are robust to outliers. It centers and scales the data based on the interquartile range (IQR), making it suitable for datasets with outliers.
- StandardScaler: Scales features to have a mean of 0 and a standard deviation of 1, making the features more comparable. It is sensitive to outliers but works well with normally distributed data.
- MinMaxScaler: Scales feature a specified range, typically between 0 and 1. It preserves the shape of the original distribution and is less affected by outliers.

a) Hyperparameter Tuning: For hyperparameter tuning, we optimize parameters like alpha (regularization parameter) and penalty (type of regularization) using grid search and StratifiedKFold cross-validation. We configure the class_weight parameter to "balanced" to address the class imbalance. The parameter grid for SGD Logistic Regression is defined as follows:

```
parameter_grid = {
    'alpha': [10**-5, 10**-2, 10**1],
    'penalty': ['l1', 'l2']
}
```

*4) **XGBoost**:* XGBoost is an ensemble learning method that combines multiple weak predictive models (decision trees) to create a more robust and accurate model. It utilizes gradient boosting to iteratively improve the model's performance.

a) Hyperparameter Tuning: For hyperparameter tuning of XGBoost, we adjust parameters including max_depth (tree Depth), min_child_weight (child node weight), learning_rate, max_bin, and n_estimators (number of Trees) using *RandomizedSearchCV* method and Stratified K-Fold cross-validation.

```
parameter_grid = {
    'max_depth':range(3,10,2),
     'min_child_weight':range(1,6,2),
     'learning_rate': [0.30000012],
     'max_bin':[256],
     'n_estimators':[100, 200]
}
```

*5) **Artificial Neural Network**:* Similar to logistic regression, scaling is also required for ANN. Our ANN model is structured as a sequential model, consisting of three dense layers and a ReLU activation function, interspersed with dropout layers to reduce overfitting by randomly deactivating a subset of neurons during training. The final output layer uses a sigmoid activation to produce a probability output, and the model is compiled with the Adam optimizer and uses binary cross-entropy as the loss function.

a) Hyperparameter Tuning: The parameters we tuned for ANN are units (number of neurons per layer), dropout rate, and learning rate (speed of model learning adjustments), and the grid is defined as follows:

```
parameter_grid = {
     'units': [32, 64, 128],
     'dropout_rate':  [0.1, 0.2, 0.3, 0.5],
     'learning_rate':[0.01, 0.001, 0.0001]
}
```

IV. RESULTS AND COMPARATIVE ANALYSIS

We systematically assess the performance of our predictive models, utilizing a selected suite of evaluation metrics tailored to the nuances of our loan classification with unbalanced datasets. We then highlight the results of our best-performing model, delve into a comparative analysis of all models considered, and discuss considerations for real-world application to ensure the models' practical relevance and effectiveness in operational settings

A. Evaluation Metrics and Logic

Given the highly unbalanced nature of our dataset, traditional metrics such as Accuracy and Area under the ROC Curve can be misleading as they tend to favor the majority class. Therefore, we opt for a set of evaluation metrics tailored to assess the performance of predicting loan defaults within an unbalanced dataset. The prioritized evaluation metrics are as follows:

1) **AUPRC**: This metric evaluates the relationship between precision (the proportion of true positives among all predicted positives) and recall (the proportion of true positives detected overall actual positives) across different thresholds. A higher AUPRC value indicates better performance, particularly important in datasets where the positive class (defaults) is less frequent.
   Importance: AUPRC is most critical to us because it offers an integral evaluation that encapsulates both precision and recall, particularly for unbalanced datasets. This measure is instrumental in our analysis as it underscores the model's proficiency in accurately identifying true defaults while maintaining a critical balance with the correct recognition of non-default cases, a crucial aspect in the domain of credit risk assessment
2) ***'Charged Off'* Recall Percentage**: This measures the recall for the *'Charged Off'* category, calculated as the ratio of correctly predicted *'Charged Off'* loans (defaults) to the total actual *'Charged Off'* loans. High recall is crucial for lenders as it reduces the risk of approving a bad loan.
   Importance: Recall is prioritized next since identifying most defaults correctly (high sensitivity) is crucial to prevent financial losses, even at the expense of increased false positives.
3) ***'Charged Off'* F1 Score**: The F1 score is the harmonic mean of precision and recall, focusing on the *'Charged Off'* category. It balances both precision and recall, which is useful when seeking a model that performs well under both criteria in an unbalanced dataset.
   Importance: The F1 score ranks next because it provides a balance between recall and precision, important for evaluating the overall efficiency of the model in managing trade-offs between catching defaults and not misclassifying good loans.
4) ***'Charged Off'* Precision Percentage**: This metric represents the precision for the *'Charged Off'* category, calculated as the proportion of true *'Charged Off'* predictions to all predictions labeled as *'Charged Off'*. High precision ensures that when a model predicts a loan will default, it is very likely to be correct, minimizing false alarms.
   Importance: Precision is important to ensure that the financial institution can maintain trust and relationships with good borrowers by not incorrectly classifying their loans as likely to default.

B. Model Results

1) Effects of Scaling on Logistic Regression and ANN Models: From the results observed in our Logistic Regression and ANN models, as documented in Appendix, Table A.I and illustrated in their Precision-Recall Curves shown in Appendix A, Figure A.I and Figure A.II, the performance differences among the models using three different scalers are minimal. Consequently, we optimized our baseline model using Minmax scaling for both algorithms.

2) Overall Model Performance and Results: Table I above presents the best results of each model. The comprehensive details for all models created with each algorithm are available in Appendix A, Table A.I.

As indicated in Table I and Table A.1, all complex models surpass our Naïve Bayes baseline model in both *'Charged Off'* F1 score and AUPRC, which are recorded at 0.45 and 0.56, respectively. The significance of surpassing the baseline model is crucial as it demonstrates the effectiveness of more sophisticated algorithms in handling the complexities of unbalanced datasets. These improvements are indicative of enhanced predictive performance and a better understanding of the underlying patterns that lead to loan defaults. This aligns with our understanding that Naïve Bayes, which assumes strong independence between its features, is often outperformed by other models that can better account for interactions among features.

We find that the Average Precision Score serves as a superior scoring metric compared to the F1 Score across all experimental models. This can be attributed to the fact that the Average Precision Score comprehensively considers the entire spectrum of precision-recall trade-offs, offering a holistic overview of model performance across various thresholds. On the other hand, the F1 Score concentrates solely on a single point on the precision-recall curve, potentially overlooking the overall performance effectiveness.

Employing the evaluation sequence described previously, we observe that our best-performing model is the optimized XGBoost model. This model utilizes `scale_pos_weight` to adjust for the imbalance in the training dataset and employs the Average Precision Score as its scoring metric.

C. Best Model Performance and Results

Table II summarizes the performance of our best model, while Figures 4 and 5 display the confusion matrix and AUPRC for our optimized XGBoost model, respectively.

D. Comparative Analysis

Among the five models evaluated for our binary classification problem—to predict loan defaults—the XGBoost model emerges as the top performer, closely followed by Random Forest. Both models are ensembles of decision trees, which is a method known for its

TABLE I: Comparison of Model Performances

Models	Description	Precision		Recall		F1-score		AUPRC
	Training Data / Scoring Metric	Fully Paid	Charged Off	Fully Paid	Charged Off	Fully Paid	Charged Off	Charged Off
Naïve Bayes	Entire Dataset / F1-weighted OR APS	0.86	0.72	0.97	0.33	0.91	0.45	0.56
Random Forest	Undersampling / APS	0.98	0.5	0.83	0.8	0.9	0.61	0.77
Logistic Regression	Entire Dataset / APS	0.94	0.52	0.82	0.77	0.88	0.62	0.77
XGBoost	Entire Dataset / APS	0.94	0.51	0.81	0.8	0.87	0.62	0.78
ANN	Undersampling / APS	0.93	0.51	0.82	0.75	0.87	0.61	0.76

TABLE II: Performance of Best Model: Optimised XGBoost

	Precision		Recall		F1-score		AUC	
Model/ (Description)	Fully Paid	Charged Off	Fully Paid	Charged Off	Fully Paid	Charged Off	PRC	ROC
XGBoost/ (Entire Dataset/ APS)	0.94	0.50	0.80	0.80	0.87	0.62	0.78	0.91

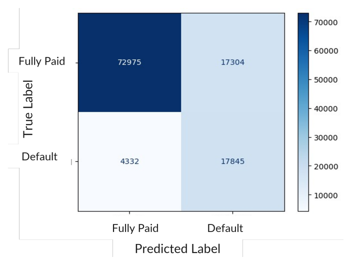

Fig. 4: Confusion Matrix of Best Model

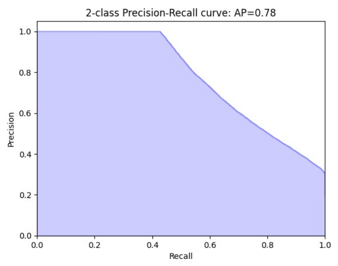

Fig. 5: AUPRC of Best Model

robust performance in various machine learning tasks, including binary classification.

Reasons for XGBoost's Good Performance:

1) Imbalanced Data Handling: XGBoost effectively tackles class imbalance through the *scale_pos_weight* parameter, prioritizing the minority class (defaults) crucial for financial risk prediction.
2) Regularization: Built-in regularization techniques prevent overfitting, ensuring the model generalizes well to unseen data, especially valuable for complex financial datasets.
3) Balanced Optimization: Using the Average Precision Score optimizes for both precision and recall, making it ideal for the imbalanced dataset where accuracy alone wouldn't be sufficient.

Similar to XGBoost, Random Forest is also an ensemble of decision trees and therefore inherits many of the benefits of this approach. It performs well in binary classification tasks, particularly on unbalanced datasets.

Reasons for Random Forest's Good Performance:

1) Robustness through Undersampling: Random Forest was applied with undersampling to even out the class distribution during training, reducing the model's bias towards the majority class. This method enhances the model's ability to learn the characteristics of the minority class, improving its accuracy in predicting 'Charged Off' loans.
2) Class Weight Balancing: Similarly, our best Random Forest model uses the `class_weight = 'balanced'` setting. This adjustment helps to enhance the model's sensitivity to the minority class, further reducing bias and improving overall classification performance on imbalanced datasets.

It is worth noting that we recognize the Artificial Neural Network (ANN) as a powerful model, adept at modeling complex, non-linear relationships and interactions between variables. Given the intricate nature of our dataset, which is not easily defined by linear decision boundaries, ANNs should theoretically excel. However, due to time and computational constraints, we could not fully optimize our ANN model to potentially surpass our other models. We believe that with more investment in training and the integration of techniques like regularization and early stopping, there is a strong potential to develop a more robust ANN model. Such a model could outperform established methods like XGBoost or Random Forest, especially with precise tuning and adequate computational resources to support the deep learning process.

E. Real-World Application Consideration

Regarding Table II, our best model achieves an 80% recall rate for the 'Charged Off' category, accurately identifying 80% of actual defaults. Its 50% precision indicates that half of the predicted defaults are correct. This efficiency makes the model an ideal preliminary screening tool for financial institutions to identify high-risk customers. Further detailed analysis on this group helps ensure that creditworthy clients aren't misclassified, enhancing the lending process by reducing default risks and maintaining customer trust.

To improve the model's real-world application, we could train it with more default data to boost precision. Additionally, techniques like stacking or model sequencing could enhance both recall and precision. These methods use multiple models in sequence, leveraging their strengths to offset weaknesses and improve overall performance. This approach gradually refines predictions, enhancing decision-making reliability.

V. CONCLUSION

This project developed a machine learning model to predict loan defaults in the banking sector, evaluating various optimization techniques. XGBoost proved most effective, providing banks a robust tool for proactive risk management and informed lending decisions. Future work may improve the model by implementing advanced data preprocessing methods like Weight of Evidence (WoE) and Information Value (IV), and by analyzing errors across loan types and demographics to enhance prediction accuracy and reliability.

References

[1] I O Eweoya et al 2019 J. Phys.: Conf. Ser. 1299 012038

[2] Malekipirbazari M, Aksakalli V Risk assessment in social lending via random forests[J]

[3] X. Ma, J. Sha, D. Wang, Y. Yu, Q. Yang, X. Niu, Study on A Prediction of P2P Network Loan Default Based on the Machine Learning LightGBM and XGboost Algorithms according to Different High Dimensional Data Cleaning, Electronic Commerce Research and Applications (2018), doi:https://doi.org/10.1016/j.elerap.2018.08.002

[4] Brown I., Mues C.An experimental comparison of classification algorithms for imbalanced credit scoring data sets Expert Systems with Applications, 39 (2012), pp. 3446-3453

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Breiman L.Random forests Machine Learning, 45 (1) (2001), pp. 5-32

[7] Addo P M, Guegan D and Hassani B 2018 Credit risk analysis using machine and deep learning models Risks 6 p 38

[8] Vaidya A 2017 Predictive and probabilistic approach using logistic regression: application to prediction of loan approval The 8th Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT) 1 pp 1–6

[9] Hamid A J and Ahmed T M 2016 Developing prediction model of loan risk in banks using data mining Machine Learning and Applications: An Int. Journal (MLAIJ) 3 pp 1–9

[10] Siddharth Vinod Jain, Manoj Jayabalan, "Applying Machine Learning Methods for Credit Card Payment Default Prediction With Cost Savings", Biomedical and Business Applications Using Artificial Neural Networks and Machine Learning, pp.285, 2022.

APPENDIX A

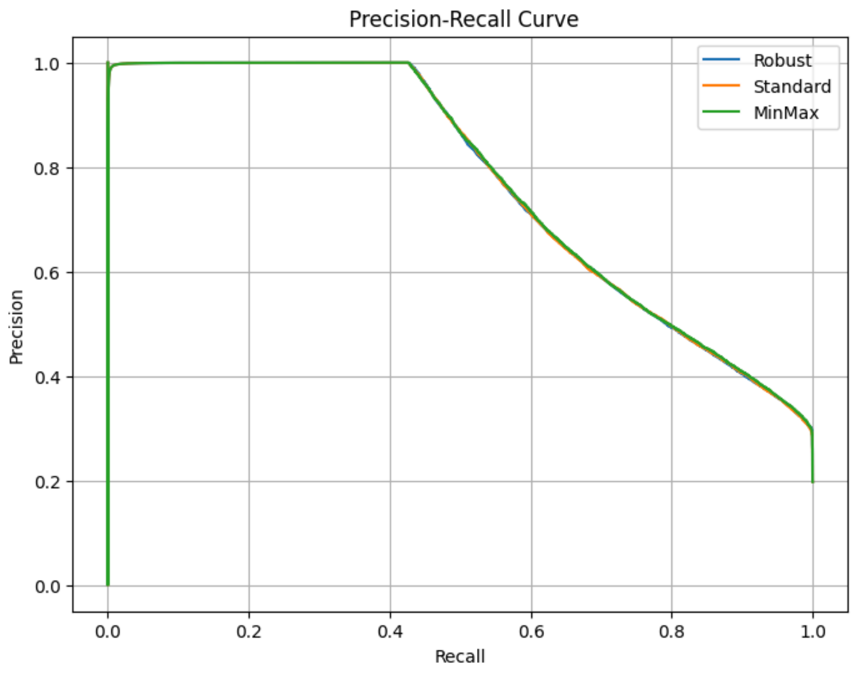

Figure A.I: Logistic Regression Precision-Recall: Robust vs. Standard vs. Min-Max

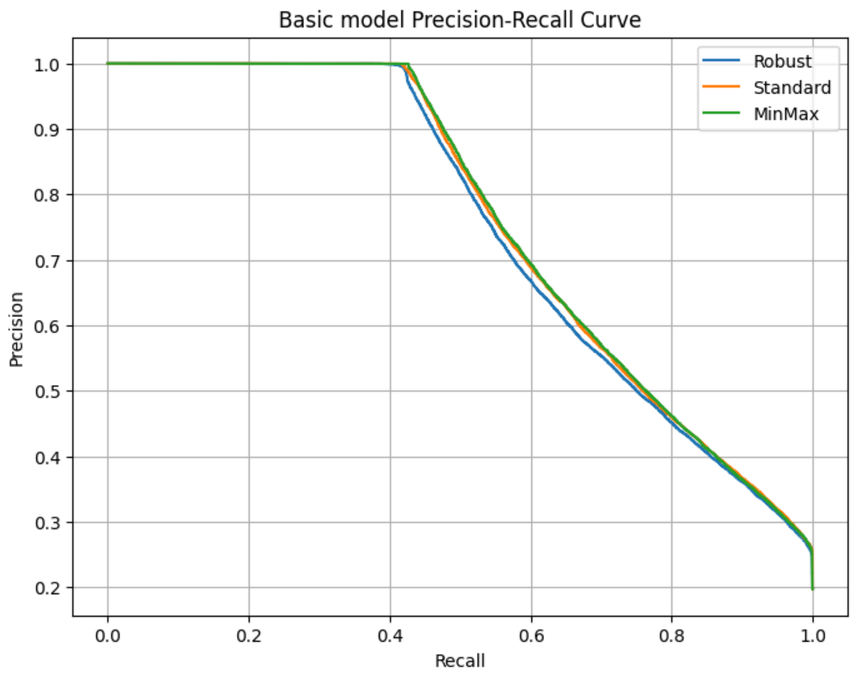

Figure A.II: ANN Precision-Recall: Robust vs. Standard vs. Min-Max

Table A.I: Performance Comparison of All Models Tested

Machine Learning Models	Optimised Models	Scoring Metric	Precision		Recall		F1-score		AUC: Precision-Recall
			Fully Paid	Charged Off	Fully Paid	Charged Off	Fully Paid	Charged Off	
Naïve Bayes	Base Model		0.87	0.55	0.91	0.44	0.89	0.49	0.53
	Entire Training Dataset	F1-weighted score	0.86	0.72	0.97	0.33	0.91	0.45	0.56
		Average Precision Score	0.86	0.72	0.97	0.33	0.91	0.45	0.56
	Undersampling	F1 score	0.9	0.37	0.73	0.67	0.8	0.48	0.53
		Average Precision Score	0.9	0.37	0.73	0.65	0.8	0.47	0.54
Random Forest	Base Model		0.88	0.94	0.99	0.44	0.93	0.6	0.75
	Entire Training Dataset (class_weight = ‘balanced’)	F1-weighted score	0.91	0.69	0.93	0.62	0.92	0.65	0.77
		Average Precision Score	0.94	0.51	0.81	0.8	0.87	0.63	0.76
	Undersampling	F1 score	0.94	0.50	0.81	0.79	0.87	0.61	0.77
		Average Precision Score	0.98	0.50	0.83	0.8	0.9	0.61	0.77
Logistic Regression	Base Models (Robust Scaler)		0.88	1.00	1.00	0.43	0.93	0.6	
	Base Models (Standard Scaler)		0.88	0.93	0.99	0.46	0.93	0.62	
	Base Models (MinMax Scaler)		0.88	0.95	0.99	0.45	0.93	0.62	
	Entire Training Dataset	F1-weighted score	0.91	0.53	0.86	0.67	0.88	0.59	0.73

	(class_weight = ‘balanced’)	**Average Precision Score**	0.92	0.61	0.89	0.69	0.91	0.64	0.78
	Undersampling	**F1 score**	0.94	0.52	0.82	0.77	0.88	0.62	0.77
		Average Precision Score	0.93	0.53	0.84	0.76	0.88	0.62	0.77
XGBoost	**Base model**		0.89	0.9	0.99	0.48	0.93	0.63	0.78
	Entire Training Dataset (‘scale_pos_weight’)	**F1-weighted score**	0.92	0.58	0.88	0.68	0.9	0.63	0.76
		Average Precision Score	0.94	0.51	0.81	0.8	0.87	0.62	0.78
	Undersampling	**F1 score**	0.94	0.50	0.8	0.8	0.87	0.61	0.78
		Average Precision Score	0.94	0.50	0.8	0.8	0.87	0.62	0.78
ANN	**Base Model (Robust Scaler)**		0.89	0.83	0.97	0.5	0.93	0.62	0.75
	Base Model (Standard Scaler)		0.88	0.98	1.00	0.43	0.93	0.6	0.76
	Base Model (MinMax Scaler)		0.88	0.96	1.00	0.45	0.93	0.61	0.76
	Entire Training Dataset	**F1 score**	0.88	0.94	0.99	0.46	0.93	0.62	0.76
		Average Precision Score	0.88	0.98	1.00	0.44	0.93	0.6	0.76
	Undersampling	**F1 score**	0.92	0.55	0.86	0.71	0.89	0.62	0.75
		Average Precision Score	0.93	0.51	0.82	0.75	0.87	0.61	0.76