Credit Approval Prediction

Xiaoqian Wang, Monan Lu

1 Introduction

In financial services, accurately predicting whether a person will be a reliable borrower is crucial for risk management and maintaining customer satisfaction. Traditional credit scoring methods are effective but integrating modern machine learning techniques can enhance both accuracy and efficiency. This project focuses on refining the decision-making process for identifying individuals with good or bad credit behavior, ensuring precise evaluations. The approval processes depend on several factors including a borrower's credit history, income stability... Our goal is to develop a machine-learning model that swiftly and accurately distinguishes between potential good and bad borrowers, reducing default risks and benefiting both lenders and borrowers to different financial scenarios.

Data overview

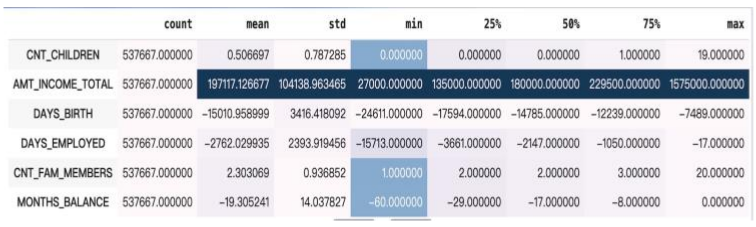
The dataset was sourced from Kaggle(https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction/data) and consists of two primary tables. The first table, named "Application Record," comprises individual client information typically used by financial institutions for credit scoring or personal loan applications. The second table, known as "Credit Record," includes financial history data for clients. Integration of these two datasets is facilitated through the use of a unique ID field present in both tables. This integration allows for the enhancement of predictive modeling efforts by combining demographic and behavioral data. The dataset encompasses approximately 540,000 observations, making it a substantial resource for analysis.

The dataset features a diverse array of data types, including binary indicators, categorical variables, and continuous numerical values. A detailed description of each feature within the dataset is provided, with the 'status' variable designated as the target variable that the model aims to predict. The remaining variables serve as input features for training purposes.

2 Data Preprocessing

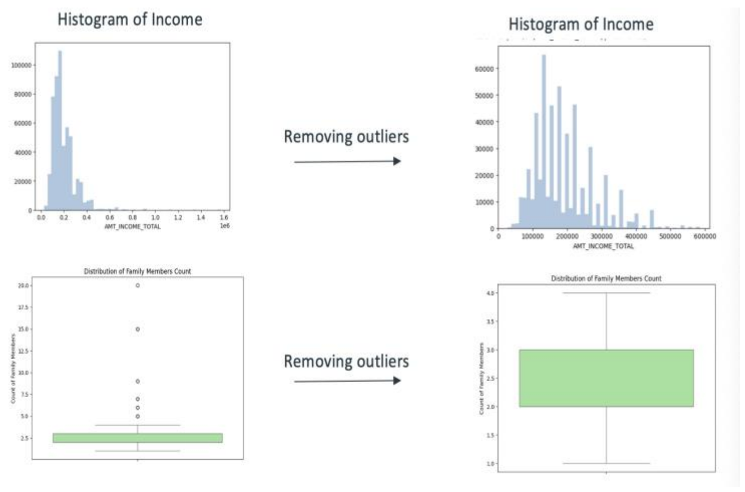
2.1 Numerical analysis

A statistical summary table of numerical data is to check whether there are potential outliers that influence the model prediction.

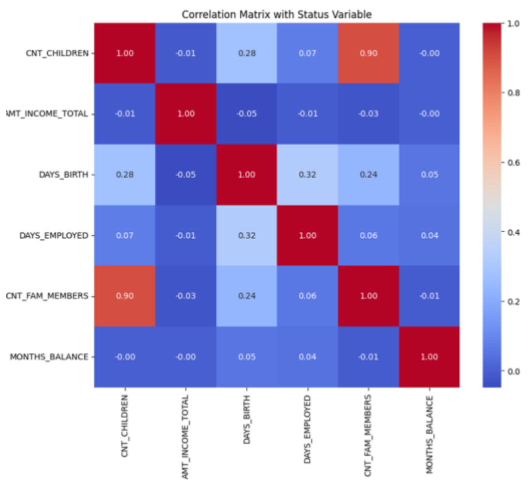


	count	mean	std	min	25%	50%	75%	max
CNT_CHILDREN	537667.000000	0.506697	0.787285	0.000000	0.000000	0.000000	1.000000	19.000000
AMT_INCOME_TOTAL	537667.000000	197117.126677	104138.963465	27000.000000	135000.000000	180000.000000	229500.000000	1575000.000000
DAYS_BIRTH	537667.000000	−15010.958999	3416.418092	−24611.000000	−17594.000000	−14785.000000	−12239.000000	−7489.000000
DAYS_EMPLOYED	537667.000000	−2762.029935	2393.919456	−15713.000000	−3661.000000	−2147.000000	−1050.000000	−17.000000
CNT_FAM_MEMBERS	537667.000000	2.303069	0.936852	1.000000	2.000000	2.000000	3.000000	20.000000
MONTHS_BALANCE	537667.000000	−19.305241	14.037827	−60.000000	−29.000000	−17.000000	−8.000000	0.000000

From the summary, the features ‘AMT_INCOME_TOTAL’and‘CNT_FAM_MEMEBRS’ both have a large gap between the minimum and maximum, Specifically, we will remove the outliers by observing histogram and boxplot like in bellowed Figure.



The correlation matrix was applied in next procedure to check the correlation between each numerical feature, ranging from -1(perfect negative correlation) and +1(perfect positive correlation).



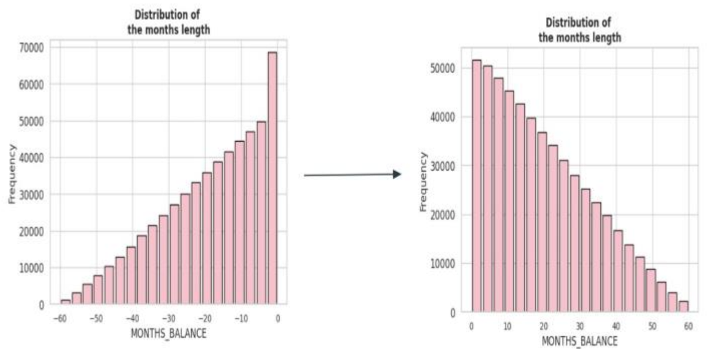
The number of children, and number of family members have a large positive correlation of 0.9. The feature 'CNT_CHILDREN' was removed to avoid multicollinearity.

Missing value

Missing data in the dataset is also a common issue particularly in the financial sector where not all information may be reported or collected consistently. For the features that have fewer missing rows, we simply removed them; for those have large numbers with missing values, we replaced them with impute the mean value of the feature. We have around 10,000 missing values for "total income", counting a small proportion, so all of them will be removed.

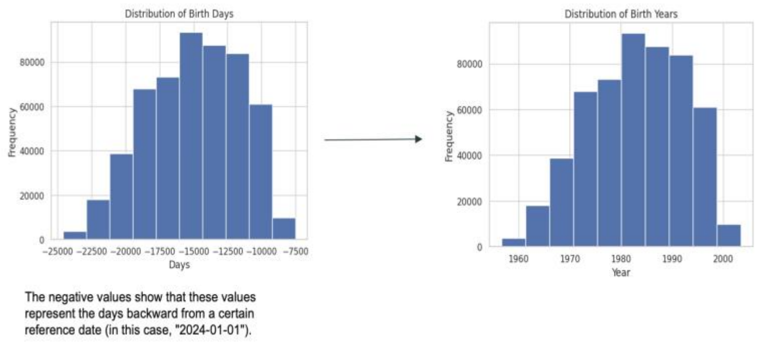
Feature engineering

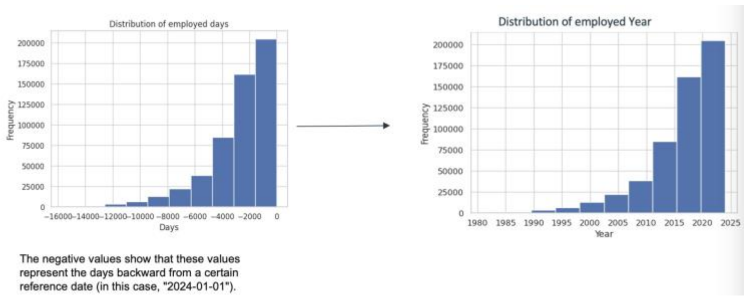
The feature: 'month balance', indicating months backward from a certain reference date, with 0 being the reference month (e.g., January 2024), and negative values showing months prior to January 2024. This is typical in financial data where transactions or balances are tracked over time relative to a specific starting point. We transform these values by -1. Transforming these values helps in making the data more understandable, especially when converts past references into a forward-looking timeline.



Negative values **Positive values**

For the feature "Birth Days," the negative values represent the age of individuals in days, counted backwards from a reference date. For instance, -10,000 days would correspond to approximately 27 years before 2024. To provide demographic analysis that provides age-related insights, we first converted "Birth Days" into exact years. Then, using 2024 as the reference date, we minused this calculated age to determine the precise birth years of the individuals. We applied the same method to the "Days Employed" feature.





2.2 Categorical analysis

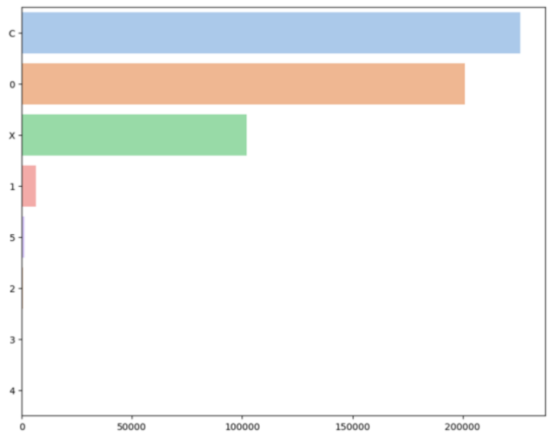
Missing value

We dealt with the missing value like the numerical data. Similarly, we have around 20,000 missing values in

"Occupation", which was a small component of the entire data, all of them were removed.

Feature engineering

Histogram of status



The feature 'STATUS' is our target feature for prediction, includes 0: 1-29 days past due, 1: 30-59 days past due, 2: 60-89 days overdue, 3: 90-119 days overdue, 4: 120-149 days overdue, 5: Overdue or bad debts, write-offs for more than 150 days, C: paid off that month and X: No loan for the month. According to most credit card websites, delinquency for 30-59 won't hurt credit too badly. Hence, we binarize "status" into two classifications: "C", "X", "0", "1" are replaced with "Good_Debt"(which represent statuses where the debt is in a good standing or paid off). And the "2", "3", "4", "5" are replaced with "Bad_Debt"(which indicate the increasing levels of risk or delinquency).

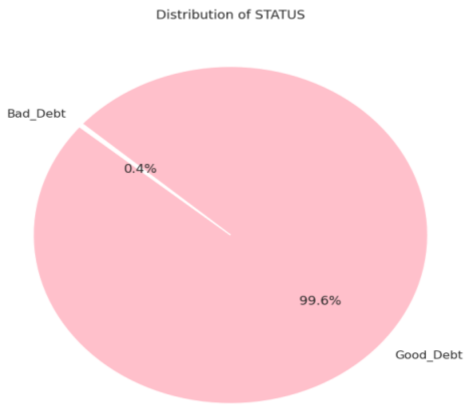
One-hot encoding

One-hot encoding was applied to handle with categorical feature to improve model performance by providing more information to the model. For each unique category in the feature, one-hot encoding creates a new binary column, which takes the value of 1 where the category is present and 0 where it is not.

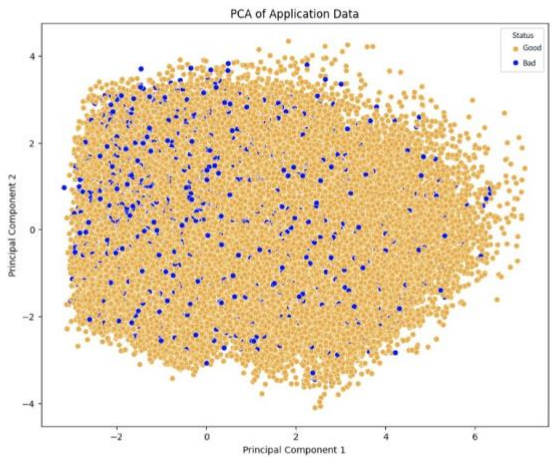
Training-testing split

Then we split the data into 80% training set, and 20% testing set for accuracy training.

Unbalanced data



The result shows that the classes in the target variable are not represented equally. Generally, Financial institutions prefer to identify someone with good credit who has been mistakenly classified as having bad credit, rather than classifying someone with bad credit as having good credit. Larger data in good debit will lead to poor generalization performance on minority class classes.



In PCA plot, the blue points are fewer in number and are scattered among the orange points, showing that the data is unbalanced.

To solve this problem, a method called "SMOTE" to create a balanced dataset by increasing the amount of data in the minority (Bad_debit) class [1]. This technique helps to balance the dataset by creating synthetic instances rather than by oversampling with replacement. It works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are then added between the chosen point and its neighbors [1]. After applying the SMOTE method, the numbers of data for 'Good_Debt' and 'Bad_Debt' are both 516973, showing that they are balanced.

3 Model selection

Based on the distribution of our dataset and the large size of the data, four models were used to make predictions.

a. Logistic Regression Model provides a relatively simple and highly interpretable model, showing the probability of class membership, and it is ideal for binary classification tasks.

The logistic model is defined as $logit\left(P(Y = 1 \mid X)\right) = \log\left(\frac{P(Y = 1|X)}{1-P(Y = 1|X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots$,

where $P(Y = 1 \mid X)$ is the probability that our label "status" is good debt given our all the x features; $X_1, X_2, \ldots, X_n$ are the predictor variables; $\beta_0, \beta_1, \ \beta_3 \ldots$. are parameters of the model, estimating from the data.

The logistic function: $g(z) = \frac{1}{1+e^{-z}}$ where z=$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$,

By substituting the logit model into the logistic function, the probability that "status" = good_debt can be modeled as $P(Y = 1 \mid X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\cdots+\beta_n X_n)}}$

The parameters $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$ are estimated using maximum likelihood estimation. The goal is to find the parameter values that maximize the likelihood function, given the observed data. The likelihood function for logistic regression (assuming independent observations) is: $L(\beta) = \prod_{i=1}^{n} [P(Y_i = 1 \mid X_i)]^{Y_i} \cdot [1 - P(Y_i = 1 \mid X_i)]^{1-Y_i}$; where Y_i is "status"for "good_debt"and "bad_debt",$P(Y_i = 1 \mid X_i)$is the predicted probability of class "good_debt" for observation i given the predictX_i.

b. Decision Tree Model, which can handle complex, non-linear relationships between features and also easily capture interactions between features.

Decision trees classify our data by splitting nodes on the most informative features, we used a criterion: Gini Impurity, to determine the best splits. Starting from the root, each node is split into two (or more) child nodes based on the attribute that best separates the classes, continuing recursively until all items at a node belong to the same class or other stopping criteria (such as maximum depth or minimal node size) are met. To prevent overfitting, through pre-pruning, we set thresholds=1000 before fully developing the tree.

$Gini(D) = 1 - (p^2 + (1-p)^2)$, where p is the proportion of one class in the node

c. The third one is the Random Forest Model, which improves on the variance of individual trees based on the Decision Tree Model, and it can be used for both classification and regression.

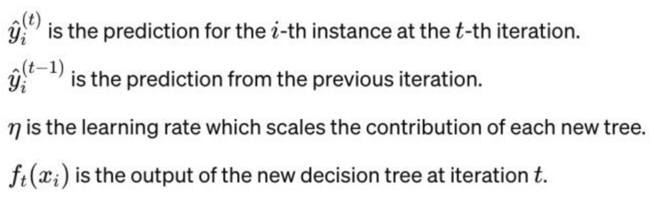
Random Forest operates by constructing a multitude of decision trees at training time and outputting the class that is the majority vote of the individual trees for our classification tasks. Each tree is built from a random sample of the data, and at each split in the tree, a subset of the features is randomly selected to determine the best split. This process of building multiple trees with randomized data and features ensures that the Random Forest model is less prone to overfitting than a single decision tree and typically delivers a significant improvement in prediction accuracy.

The prediction is the class that receives the majority of the votes from all the trees:$\hat{y} = \{f_1(x), f_2(x), \ldots, f_B(x)\}$,where $f_b(x)$is the class prediction of the b-th tree.

d. The fourth one as an extra technique is the XGBoost Model, which provides top-notch results in structured datasets and Includes L1 and L2 regularization which helps in reducing overfitting[3]. Also, it can handle large-scale and high-dimensional data[3].

XGBoost constructs the model in an additive manner:

$$\widehat{y_i^{(t)}} = \widehat{y_i^{(t-1)}} + \eta \cdot f_t(x_i)$$



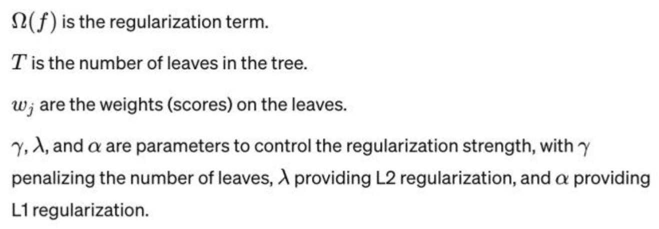
$\hat{y}_i^{(t)}$ is the prediction for the i-th instance at the t-th iteration.

$\hat{y}_i^{(t-1)}$ is the prediction from the previous iteration.

η is the learning rate which scales the contribution of each new tree.

$f_t(x_i)$ is the output of the new decision tree at iteration t.

Regularization: $\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \alpha \sum_{j=1}^{T} |w_j|$



$\Omega(f)$ is the regularization term.

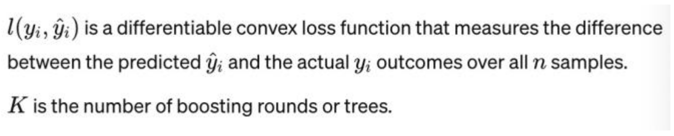
T is the number of leaves in the tree.

w_j are the weights (scores) on the leaves.

γ, λ, and α are parameters to control the regularization strength, with γ penalizing the number of leaves, λ providing L2 regularization, and α providing L1 regularization.

Combining the loss and the regularization gives the objective function that XGBoost tries to minimize:

$$\text{Obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{K} \Omega(f_t)$$



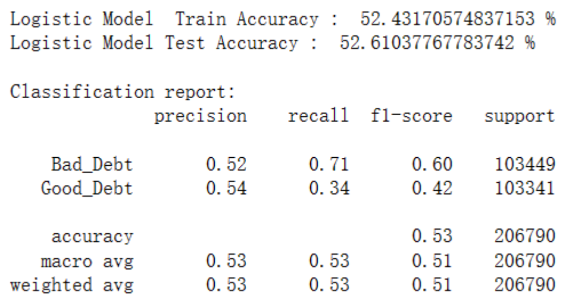
$l(y_i, \hat{y}_i)$ is a differentiable convex loss function that measures the difference between the predicted $\hat{y}_i$ and the actual y_i outcomes over all n samples.

K is the number of boosting rounds or trees.

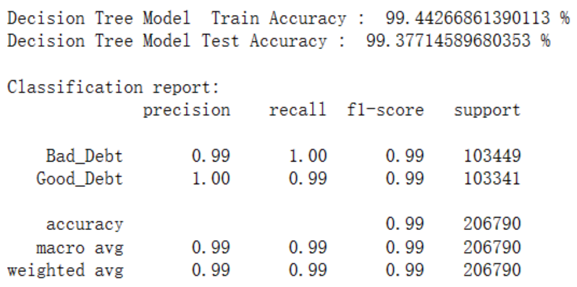
4 Model performance

By generating the accuracy on the training data, we have the classification matrix for the four models.

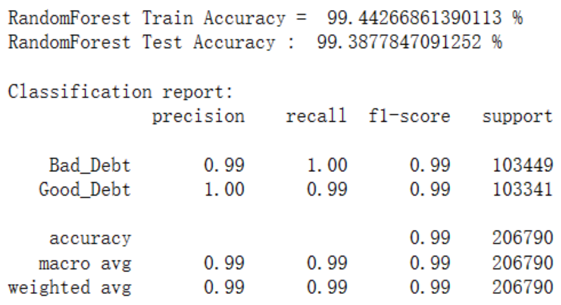
Classification Matrix



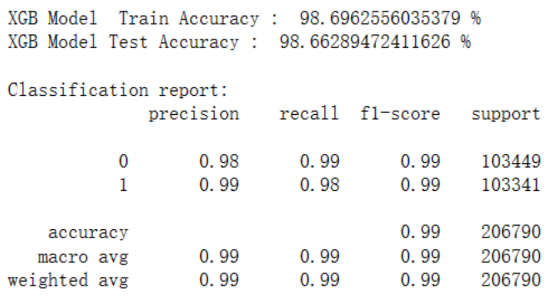
```
Logistic Model  Train Accuracy :  52.43170574837153 %
Logistic Model Test Accuracy :  52.61037767783742 %

Classification report:
              precision    recall  f1-score   support

    Bad_Debt       0.52      0.71      0.60    103449
   Good_Debt       0.54      0.34      0.42    103341

    accuracy                           0.53    206790
   macro avg       0.53      0.53      0.51    206790
weighted avg       0.53      0.53      0.51    206790
```



```
Decision Tree Model  Train Accuracy :  99.44266861390113 %
Decision Tree Model Test Accuracy :  99.37714589680353 %

Classification report:
              precision    recall  f1-score   support

    Bad_Debt       0.99      1.00      0.99    103449
   Good_Debt       1.00      0.99      0.99    103341

    accuracy                           0.99    206790
   macro avg       0.99      0.99      0.99    206790
weighted avg       0.99      0.99      0.99    206790
```



```
RandomForest Train Accuracy =  99.44266861390113 %
RandomForest Test Accuracy :  99.3877847091252 %

Classification report:
              precision    recall  f1-score   support

    Bad_Debt       0.99      1.00      0.99    103449
   Good_Debt       1.00      0.99      0.99    103341

    accuracy                           0.99    206790
   macro avg       0.99      0.99      0.99    206790
weighted avg       0.99      0.99      0.99    206790
```

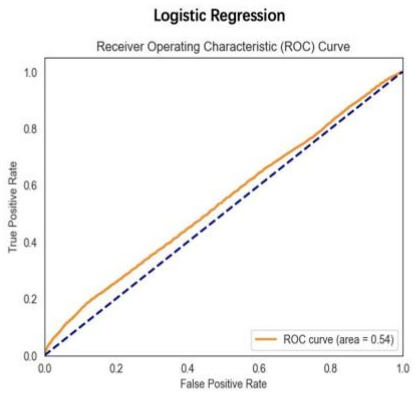


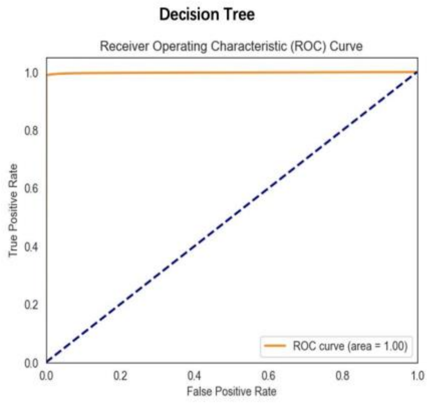
```
XGB Model  Train Accuracy :  98.6962556035379 %
XGB Model Test Accuracy :  98.66289472411626 %

Classification report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99    103449
           1       0.99      0.98      0.99    103341

    accuracy                           0.99    206790
   macro avg       0.99      0.99      0.99    206790
weighted avg       0.99      0.99      0.99    206790
```

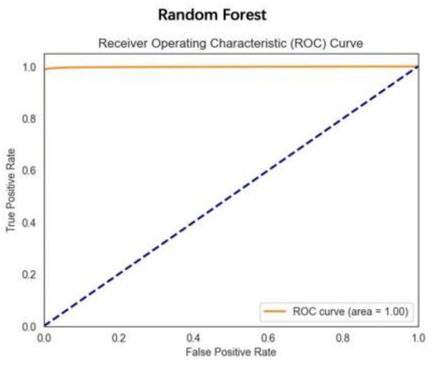
The Logistic Regression model has significantly lower performance metrics compared to the other models, with an accuracy of approximately 52.61% on the test set. The precision, recall, and F1-score for 'Bad_Debt' are modest, and particularly poor for 'Good_Debt'. This model struggles with both classes but is slightly better at identifying 'Bad_Debt'. The low F1-scores suggest that the model does not effectively balance precision and recall, which could be due to linear assumptions in the data that do not hold or insufficient feature engineering. The Decision Tree model shows a drastic improvement, boasting near-perfect accuracy, precision, recall, and F1-scores close to 1.00 across both classes. While impressive, the extremely high scores on both training and test sets might indicate overfitting despite the high test accuracy, which is a common issue with decision trees, especially if they are deep with many branches. Similar to the Decision Tree model, the Random Forest achieves very high metrics across the board, with test accuracy also around 99.38%. As an ensemble method, Random Forest typically reduces the overfitting seen in individual decision trees by averaging multiple decision trees' outcomes, which likely contributes to its robust performance. The XGBoost model also shows excellent performance, with a test accuracy of about 98.63% and high scores for precision, recall, and F1 across both classes. XGBoost is designed to optimize large gradient boosting frameworks and often performs excellently on structured data. Its slightly lower test accuracy compared to Random Forest might be due to differences in handling the ensemble learning process or specific hyperparameters.

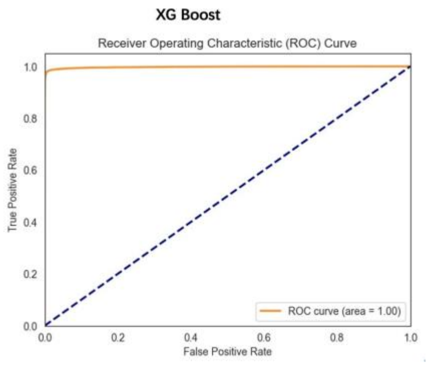
Then we generate the ROC curve, AUC and confusion Matrix for the four models.

ROC Curve



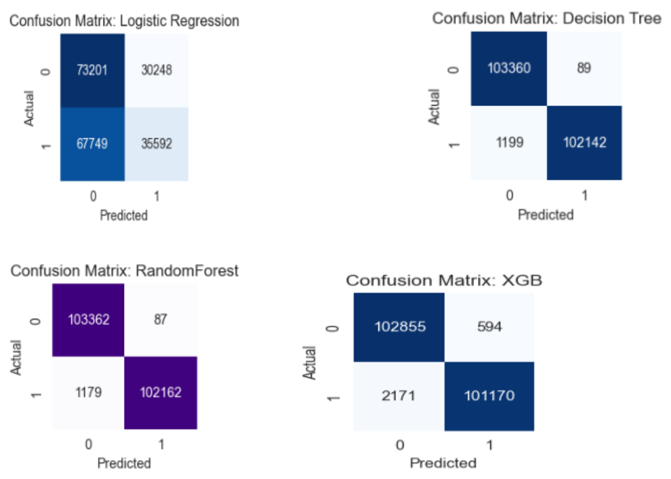






The area under the ROC curve (AUC) for the Logistic Regression is 0.54. This value is slightly above 0.5, which is only slightly better than a random guess. The ROC curve is close to the diagonal line, indicating that the model has limited ability to discriminate between the positive and negative classes. This performance is consistent with the earlier provided classification metrics and confirms that Logistic Regression is not performing well in this specific scenario. The Random Forest model shows an AUC of 1.00, indicating perfect discrimination between positive and negative classes. The ROC curve hugs the top left corner, suggesting excellent model performance. While the AUC indicates ideal performance, such a perfect score often raises concerns about overfitting, especially if the training data is not representative of the general population or if the model has memorized the training data. Like the Random Forest, the Decision Tree also shows an AUC of 1.00 with a ROC curve that reaches the top left corner. The Decision Tree's perfect score similarly suggests a risk of overfitting. It indicates that the model can perfectly classify all training instances, which may not translate well to unseen data. The XGBoost model also achieves an AUC of 1.00, displaying a ROC curve identical in performance to the Random Forest and Decision Tree. XGBoost's excellent AUC suggests that it is very effective at differentiating between the classes. However, as with the other high-performing models, care must be taken to ensure that this isn't a result of overfitting.

Confusion Matrix



Logistic Regression has a considerable number of false positives, suggesting a tendency to incorrectly predict the positive class (likely "approved" in this context). The balance between false negatives and false positives is poor, indicating the model's limited capacity to distinguish between classes effectively. The Decision Tree model shows a very high number of true positives and true negatives with very few false positives. However, there are more false negatives compared to false positives, indicating it might miss classifying some of the actual positive cases. Random Forest improves slightly over the Decision Tree in terms of both false positives and false negatives, showing a very robust performance with extremely high true positives and true negatives. This model balances sensitivity and specificity very effectively. XGBoost has more false positives and false negatives compared to the Decision Tree and Random Forest. It suggests a slight decrease in both sensitivity and specificity compared to these models, although it still performs well.

Overall Result

Decision Tree and Random Forest are the strongest performers based on the classification matrix, ROC curve and AUC, confusion matrices, but careful validation is necessary to confirm if their performance is genuinely robust or if overfitting is inflating their effectiveness. Logistic Regression may require a review of feature selection, model tuning, or even reconsideration of the model choice to improve its predictive accuracy. XGBoost offers a good balance but might need some tuning to reduce both types of errors.

5 Limitation

The near-perfect scores reported by the Decision Tree and Random Forest models, along with the Logistic Regression's poor performance, raise concerns about overfitting. Overfitting occurs when a model is too closely fitted to the limited data it was trained on, which may not generalize well to unseen, real-world data. The high scores could potentially be misleading if the models have simply memorized the training data rather than learning to generalize from it.

Also, a perfect AUC can suggest that the model may not perform as well when exposed to new data. This is especially critical if the model has been trained on a dataset that isn't fully representative of real-world scenarios. Ensuring the models are reliable, robust, and truly reflective of real-world conditions is crucial before they can be used to influence critical business decisions in a company or enterprise. We would not recommend applying these models directly into production without any further adjustment steps.

6 Future Analysis

For future analysis, the Decision Tree and Random Forest models have shown excellent performance on the given dataset, but due to the potential risk of overfitting, it is crucial to validate these models further with new, external datasets that mirror real-world conditions. This would help to ensure that the models are robust and can generalize well beyond the specific scenarios they were trained on.

SVM model and Neural Network Model may also apply to make predictions for credit approval. SVM is an effective model in high dimensional spaces, and it uses a subset of training points in the decision function. It has a good ability to generalize well and prevent overfitting effectively. But it also may face the issue of choosing, and tuning the kernel type which can be complex. And it scales poorly with large datasets. The performance is not very good when the data set has more noise. And SVM models are not very intuitive and are difficult to interpret compared to other algorithms. Neural Network Model captures complex data relationships and patterns effectively, especially with large datasets. And it learns almost any function, making them incredibly flexible and powerful. However, it may require a lot of data preprocessing and tuning. And it uses a lot of computing resources and lacks clear interpretability.

7 Fairness

Our project acknowledges the importance of fairness in machine learning applications, particularly in sensitive areas such as credit scoring, we have not yet implemented specific fairness metrics or bias mitigation strategies in our model. Moving forward, it will be crucial to integrate fairness considerations into our model development process. This will involve conducting a thorough bias audit of our training data, employing fairness-enhancing techniques such as re-sampling or re-weighting our dataset, and continuously monitoring the model's decisions to ensure they do not disproportionately disadvantage any particular group. Implementing these steps will help us achieve not only high accuracy but also fairness in our credit approval predictions, aligning our project with best practices in ethical AI.

8 Conclusion

This project has successfully demonstrated the application of advanced machine learning models to enhance the decision-making process in credit scoring. By integrating Logistic Regression, Decision Trees, Random Forest, and XGBoost models, we have developed a robust system that can accurately differentiate between potential good and bad borrowers. Despite challenges such as potential overfitting and the need for further validation, the results so far are promising, showing high accuracy and efficiency in processing large datasets.

Generally, our project lays a strong foundation for using machine learning to improve credit approval processes, with a clear path forward for enhancing model performance and fairness in future implementations. We are committed to continuing our work to bridge the gap between technical feasibility and practical, ethical application in financial services.

Reference

[1] Brownlee, J. (2021, March 16). *Smote for imbalanced classification with python*. MachineLearningMastery.com. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[2] *What is XGBoost?*. NVIDIA Data Science Glossary. (n.d.). https://www.nvidia.com/en-us/glossary/xgboost/

[3]"Credit Card Approval Prediction." *Kaggle*,

www.kaggle.com/datasets/rikdifos/credit-ca