Cornell University

ORIE 5741 Project Report

Identifying Key Predictors of Loan Default: Enhancing the Loan Approval Process through Machine Learning Analysis

Yuanzhi Ma(ym533)

Xiaotong Li(xl939)

Jiachen Ma(jm2833)

Catalogue

1. Introduction

1.1 Background of the Problem

Our goal is to analyze historical loan data to identify key predictors of loan default, categorized into statuses like Fully Paid, Charged Off, Default, and Late. By understanding these predictors, we aim to improve the loan approval process, enhancing decision-making and minimizing financial risk for lending institutions. This analysis will allow us to optimize lending practices by balancing risk management with the accessibility of credit.

1.2 Describe the Data Set

The dataset named "Credit Risk Analysis" obtained from Kaggle comprises over 880,000 records and 20+ columns, capturing loan and credit history data from December 31, 2006, to December 29, 2015, provided by LendingClub Bank. For our project, we have filtered the dataset to include 13 specific features across 20,318 data points from New York State. Each data point in this subset corresponds to an individual loan, where every row represents a unique loan application and each column details various attributes or features related to the loan and the borrower's financial profile. These selected features aim to offer an in-depth insight into the loan dynamics and the borrower's financial status.

1.3 Significance of Problem

Predicting loan status can have a direct impact on the bank's cash flow and overall financial liquidity and stability. By accurately forecasting loan defaults, the bank not only improves its financial health but also enhances its cash flow management. This predictive insight facilitates improved risk management, leading to more informed loan approval decisions, thus optimizing the balance between risk and return making loan, also making sure to provide loans to only customers who truly need it.

Therefore, this project's focus on loan status prediction is essential for the bank's risk management strategies, operational effectiveness, and service quality, all of which are fundamental to its long-term development and contribution to the global economy. By ensuring financial stability and fostering trusted customer relationships, the project aligns with the bank's commitment to operational excellence and economic impact, underpinning its enduring success and relevance in the financial sector.

1.4 How to Solve the Problem

Our project aims to predict future loan statuses by analyzing a dataset from LendingClub Bank, focusing on identifying key predictors of loan defaults. We employed several sophisticated machine learning techniques to tackle this multi-class classification problem: First, Logistic Regression: We used Naive Bayes which serves as an initial analytical tool due to its efficiency and simplicity, offering quick insights that help in preliminary feature selection. Next, we multinomial logistic regression to model probabilities across multiple loan status categories, providing a strong baseline for comparison. Next, we implemented SVM with Softmax: We adapted the SVM to multi-class problems using the softmax function to manage and classify data into multiple categories, enhancing its ability to separate classes effectively. After that, Random Forest: By using this ensemble method, we leverage multiple decision trees to improve accuracy and robustness, benefiting from its capability to rank feature importance and control overfitting. Lastly, Deep Learning (DL): Our deep learning models capture complex patterns through tailored neural network architectures, addressing non-linear relationships and providing detailed analysis capabilities.

We then assess each model using key performance metrics like Confusion matrix, model accuracy, precision, and recall. Through these diverse and robust methodologies, we aim to enhance the predictive accuracy of loan defaults, which will allow us to refine the bank's loan approval processes and improve overall risk management strategies. This approach not only mitigates financial risks but also aids in strategic decision-making, contributing significantly to the bank's operational effectiveness and financial stability.

2. Data Cleaning and Feature Engineering

To prepare our dataset for machine learning modeling, we first performed a train/test split to separate our data into training and testing sets. This is crucial for evaluating our model's performance on unseen data, ensuring that our predictions are robust and generalize well outside of our training dataset. Here's how we conducted the split:

2.1 Handling NA Values

We handled “NA” values and categorical variables being “Other” just by removing the data points. We did this because we have a very large dataset and we only need to remove a really small number of data points, so we could assume these data are missing by random.

2.2 Handling Unbalanced data

Our target is to predict loan_status, but there are many more data points with label fully_paid than data with other labels. The could make our model prediction favoring predicting fully_paid loans with better precision and recall. Instead of trying to balance the number of data points in our data points, we handled this unbalance in our models by assigning different penalty weights for misclassifying each label.

2.3 Random Shuffling

We shuffled the dataset to randomize the order of rows. This helps in reducing variance and ensures that the training and testing sets are representative of the overall dataset.

2.4 Train-Test Split

We designated 80% of our dataset to be the training set and the remaining 20% as the testing set. The actual split was achieved by calculating the number of data points that constitute 80% of our dataset, which ensures an adequate amount of data for training while leaving out a substantial portion for testing.

2.5 Separating Features and Target

The loan_status column, which is our target variable indicating the loan's outcome, was separated from the input features. We also removed several columns (addr_state, sub_grade, funded_amnt, loan_amnt) from our feature set that were not required for the model training either due to redundancy or irrelevance to the task.

2.6 Mapping Grades

The ordinal grade variable was transformed into a numerical format using a predefined mapping scale where grades are represented as integers (with 'A' as the highest and 'G' as the lowest). This transformation allows the model to interpret this feature quantitatively.

2.7 One-Hot Encoding

We applied one-hot encoding to the nominal variables home_ownership and purpose. This technique converts categorical variables into a form that could be provided to ML algorithms to do a better job in prediction.

2.8 Scaling Features

Lastly, we scaled our features using the StandardScaler from scikit-learn. This step is essential as it normalizes the features in our dataset to a standard scale, reducing the bias that higher magnitude features impose on the model.

2.9 Summary

The training set consists of inputs train_x and targets train_y, and the testing set consists of inputs test_x and targets test_y. Both datasets are now ready for further analysis and model building. The processing steps ensure that the data fed into our models are clean, appropriately formatted, and randomized, setting a solid foundation for accurate and reliable predictive modeling.

3. Model and Algorithm Analysis

After we got a reliable dataset, we trained our prediction model using various machine learning algorithms, including Naïve Bayes, Logistic Regression, Support Vector Machine, Random Forest, and Neural Networks. Since our predicting target is nominal, we used Zero-One Loss:

$$L_{01} = \sum_{i=1}^{N} 1\{\hat{f}(x_i) \neq y_i\}$$

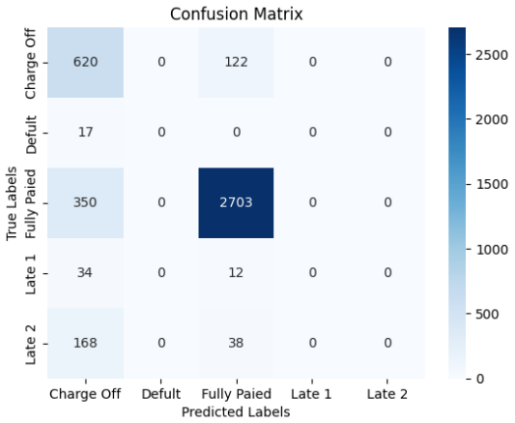
when evaluating our models. Then, we evaluated our model performance in terms of overall accuracy, number of false positive, number of false negative, precision, and recall:

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

measures how many are actually positive among the ones we predicted as positive.

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

measures how many we predicted as positive among the ones that are actually positive. We also provided confusion matrices for visualization:



3.1 Naive Bayes

3.1.1 Methodology

Naive Bayes Classifier is a probabilistic machine learning model based on Bayes' Theorem, which describes the probability of an event, based on prior knowledge of conditions that might be related to the event. The theorem is mathematically expressed as:

$$\mathrm{P(A|B)} = \frac{\mathrm{P(B|A)} \times \mathrm{P(A)}}{\mathrm{P(B)}}$$

The term "naive" refers to its fundamental assumption that, conditioned on the class label, the features in the dataset are mutually independent of each other. So, the classifier can use the following formula to make prediction:

$$\mathrm{P(C_k|x_1, x_2, \dots, x_n)} = \frac{\mathrm{P(C_k)} \cdot \prod_{i=1}^{n} \mathrm{P(x_i|C_k)}}{\mathrm{P(x_1, x_2, \dots, x_n)}}.$$

for the class label C_k of a data point with the maximum posterior probability given the input features $x_1, x_2, \dots, x_n$. Because the denominator is constant given the input, we can use the following proportionality

to make a prediction:

$$P(\mathrm{C}_k|x_1, x_2, \ldots, x_3) \propto P(\mathrm{C}_k) \cdot \prod_{i=1}^{n} P(x_i|C_k).$$

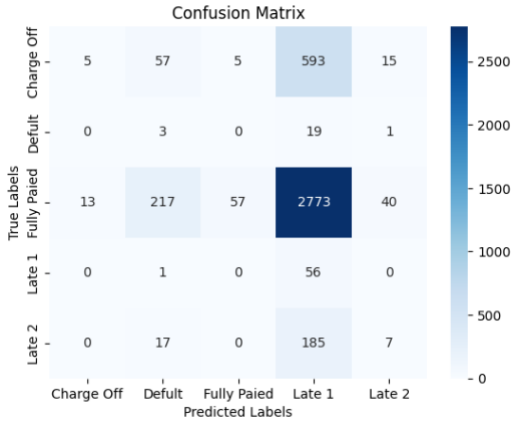
We then choose the class C_k that maximizes this probability:

$$\hat{y} = argmax_{c_k} P(C_k) \cdot \prod_{i=1}^{n} P(x_i|C_k)$$

The naïve assumption greatly simplifies the computation when handling large datasets. Although it is a strong and often unrealistic assumption, Naive Bayes classifiers can perform surprisingly well in many real-world situations.

3.1.2 Result

Confusion matrix for Naïve Bayes:



From the matrix, we can see that Naïve Bayes resulted in a suboptimal model with prediction accuracy of only 2.6%, which is exceptionally low. Since our classification task is not binary, we cannot exploit the model by choosing opposite result. This suggests that the model predictions even worse than random guessing. More specifically, we observe that the classifier was heavily biased towards the 'Late1' class but with very poor precision of about 1%. 'Charged Off' and 'Default' classes were entirely misclassified with extremely low recall of less than 10%. The reason why this happen might be that our features are highly correlated with each other even when conditioned on the label loan_status, so conditional independence is a too naïve assumption to make on our dataset.

3.2 Logistic Regression

3.2.1 Methodology

We first use Ordinary Logistic Regression (OLR), which is a discriminative algorithm which models $P(y|x)$ directly. OLR works by fitting the data to a logistic function, which allows for the estimation of the probabilities of the two possible outcomes:

$$P(y|x, w) = \frac{1}{1 + \exp(-y(x^T w))}$$

by finding the optimal weight vector w^* by solving:

$$\underset{w}{argmax}\, P(D|w) = \arg\max_{w} \prod_{i}^{N} P(y_i|x_i, w)$$

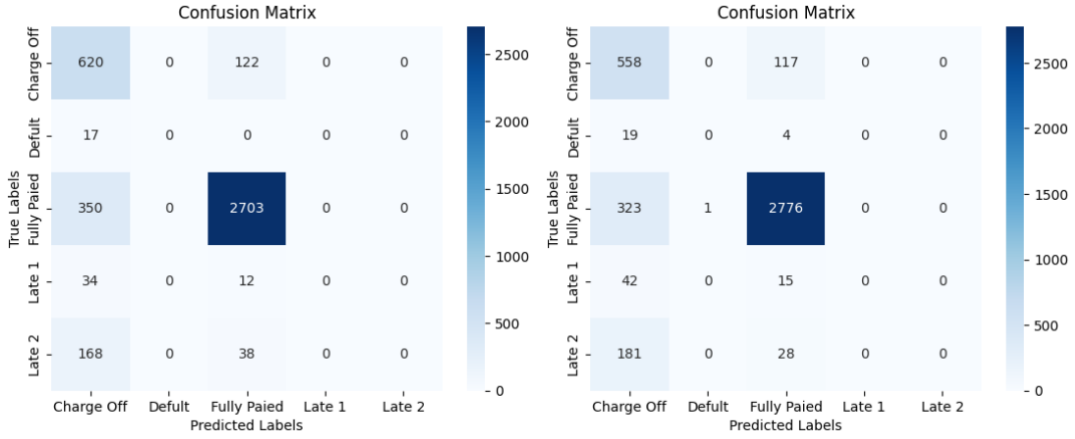
It's a go-to method for binary classification tasks due to its straightforwardness in implementation, interpretability of the coefficients. However, it can also be easily extended to multi-class classification by using the soft-max function:

$$P(y = k|x, w) = \frac{\exp(x^T w_k)}{1 + \sum_{k=0}^{K} \exp(x^T w_i)}$$

Then, we tried Improved Logistic Regression (ILG), which uses the 'lbfgs' solver, known for its efficiency with large multi-class problems, and incorporates *class_weight* to handle imbalanced data by adjusting weights inversely proportional to class frequencies. These enhancements enable the ILG to perform nuanced predictions across multiple classes, marking a significant advancement over the traditional OLR.

3.2.2 Result

Confusion Matrices for OLR (left) and ILG (right):



From the matrix, we see the OLR model achieved an overall accuracy of approximately 82%, whereas the ILG model showed an improvement with about 83% accuracy, both being effective in predicting *loan_status*. The difference in performance also indicates that the ILG model was more effective in classifying loan statuses correctly. However, the improvement is not evident by just looking at overall accuracy. The recall of 'Charge Off' 83% for OLR and 84% for ILG. Although the difference also looks small, it could mean a lot in reality, because predicting "Charge Off" as "Charge Off" is the primary objective for the Bank. Even 1% increase in this recall rate could help the bank avoid huge amount of loss.

3.3 Support Vector Machine

3.3.1 Methodology

Support Vector Machine (SVM) is a powerful classifier that work well on a wide range of classification problems, particularly when the number of features is high. They are effective in high dimensional spaces. It tries optimal w^* which maximizes the Geometric Margin between different class of data points:

$$Geometric\ Margin\ \gamma_{geom} = \frac{min_{i=1,\dots,n} y_i x_i^T w^*}{\|w^*\|}$$

conditioned on all data points are correctly classified, or allowing a soft margin with slack variable ξ_i as penalty for misclassification. The full optimization formulation is as follow:

$$\min_{w,b,\xi} \frac{1}{2}\|w\| + C\sum_{i=1}^{N} \xi_i$$

$$s.t. \quad y_i\left(x_i^T w + b\right) \geq 1 - \xi_i, \qquad \xi_i \geq 0, \qquad \forall i = 1, \dots, N$$

where C is a hyper parameter, indicating the tolerance with misclassification.

SVMs can also employ a variety of kernel functions to fit on data that is not linearly separable. For example, we can use the RBF Gaussian Kernel:

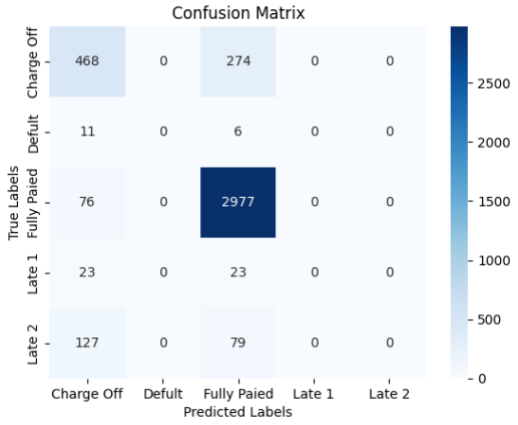
$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

to map our data points to infinitely high dimensional space, to make it separable by a linear boundary. There are many other choices of kernel function like polynomial kernel and sigmoid kernel, and the best kernel can be determined by cross-validation. In our implementation, we found the RBF kernel most effective.

While SVMs are fundamentally binary classifiers, they can be extended to handle multi-class classification problems. Using the soft-max function, we can obtain probabilities of each class output, thus determining the most likely classification.

3.3.2 Result

Confusion Matrix for SVM with RBF kernel:

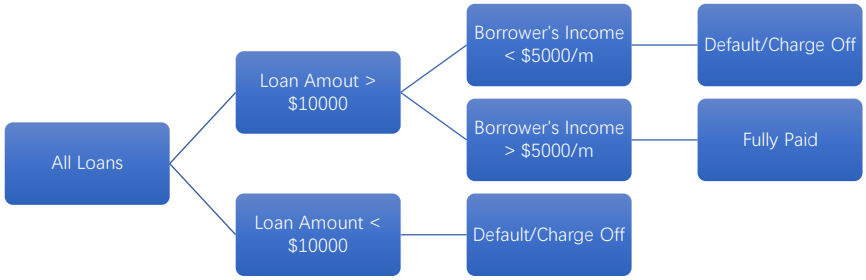


From the matrix, we see the overall classification accuracy is about 82%, which is very good for such a complicated classification task. The model has an especially high recall of 98% for the "Fully Paid" class. This means most of the "Fully Paid" loans are correctly classified as "Fully Paid", which allows most of the trustworthy clients to get loan from the bank. However, as a trade-off of high recall for "Fully Paid", it recalls for "Charge Off" is only 60%, which is highly unfavorable for the bank. That is, if the bank uses our SVM model to decide who will get a loan, 40% of loan that will eventually become "Charged Off" will be granted. This can potentially result in huge loss to the bank and may even result in liquidity crisis. To further improve the model, one method is to add more penalty to misclassifying data with label being "Charge Off".

3.4 Random Forest

3.4.1 Methodology

To understand Random Forest, we need to first know what a Decision Tree is. A Decision Tree is a multi-class classification or regression algorithm. A tree will have a root node that contains all data points, and sub nodes containing subsets of data base on certain metrics. At the end of each branch will be a leaf node containing final decision of the classification task. An example is as below



When training the decision tree, as each split, we try to minimize the diversity of class labels in each leaf node after splitting. There are many metrics for diversity, such as Entropy and Gini Impurity.

The Random Forest algorithm is an ensemble learning technique that constructs a multitude of decision trees at training time and outputs the mode of the classified classes or mean regressed prediction of the

individual trees. It also introduces randomness by selecting random samples of the dataset to build each tree and by choosing a subset of features at each split point, enhancing model robustness against overfitting and improving generalization to unseen data. This makes Random Forest a powerful and versatile classifier capable of handling complex datasets with high accuracy. The process of training a Random Forest is as following:

1. Bootstrap aggregating (Bagging): Random Forest starts by selecting random subsets of the dataset with replacement, a process known as bootstrap aggregating, or bagging. Each subset is used to train a separate decision tree.

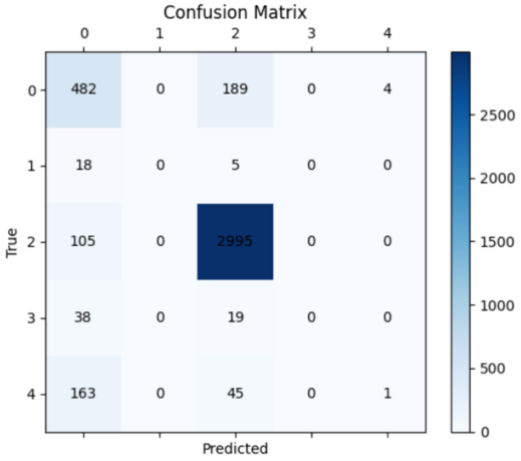
2. Feature Randomness: When splitting a node during the construction of a tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. This adds diversity to the model, making it more robust to overfitting.

3. Aggregation: After many trees are grown, the Random Forest algorithm makes a prediction by averaging the predictions of all the individual trees for regression tasks or by taking a majority vote for classification tasks.

4. Grid Searching for Hyperparameters: There are several hyperparameter in a Random Forest, including number of Trees, Maximum Depth of each Tree, number of Features to consider at each split, number of data points to Sample for each Tree during Bagging. We used cross validation to search for their optimal values.

3.4.2 Result

Confusion matrix for Random Forest:



"Fully Paid" has a very high number of True Positives (TP) at 2995, "Charge off "also has a relatively high number of TPs at 482, indicating that the Random Forest model is highly effective at identifying "Fully Paid" and "Charge off" when the data are actually in the classes. For "Charge Off", there are 193 misclassifications where it was mistaken as "Fully Paid" and other class, making its Recall for "Charge Off" being 71%. The matrix suggests a possible class imbalance, particularly for "Charge Off", which has very few instances leading to lower predictive performance. To fix this problem, we could further assign more penalty for misclassifying data with class label "Charge Off".

3.5 Deep Learning

3.5.1 Methodology

We also used a sequential neural network facilitated by Keras, a high-level neural networks API to assist us with the task. The architecture comprises three layers: an input layer with 64 neurons, a subsequent hidden layer containing 32 neurons, and an output layer consisting of 5 neurons. The choice of 64 and 32 neurons for the respective layers aims to create a model complex enough to capture the underlying patterns in the data without being too computationally expensive. The output layer employs a soft-max activation function, a standard in multi-class classification problems, which outputs a probability distribution over the five predicted

classes.

$$\text{Softmax}(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

For the model compilation, the data is normalized and passed into the NN the Adam (Adaptive Moment Estimation), a popular optimizer used in training deep learning models is used. It combines ideas from both momentum and RMSprop optimizers to handle sparse gradients on noisy problems.

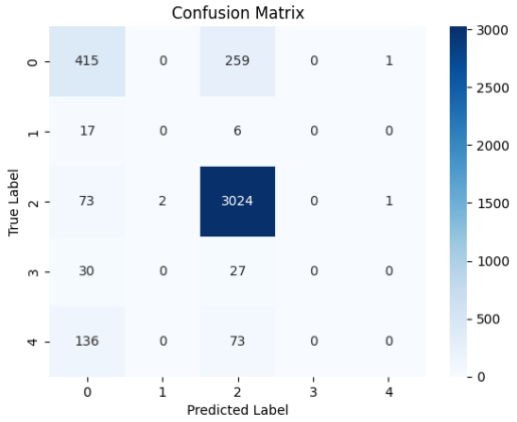
Interestingly, for classification task the most common selected the loss function specified is binary_crossentropy, which typically applies to binary classification tasks. However, for a multi-class classification, categorical_crossentropy would generally be the appropriate choice, according to Geoffrey S. H instead we've decided to use the Categorical Cross Entropy Given by the formula below:

$$\text{Categorical Cross Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\widehat{y_{i,c}})$$

Here, N represents the number of observations, C denotes the number of categories.$y_{i,c}$ is a binary indicator that equals 1 if class c is the correct classification for observation i and 0 otherwise. $\widehat{y_{i,c}}$ indicates the predicted probability that observation i is in class c.During compilation, the model's performance is evaluated based on accuracy.

The training process involves 20 epochs, allowing the model to iteratively learn from the training data. A relatively small batch size of 32 is used, which is beneficial for generalizing the model as it updates weights more frequently. The inclusion of a validation split implies that 10% of the training data is held back to evaluate the model's performance during training, which helps in monitoring overfitting.

3.5.2 Result



As shown in the confusion matrix, The model demonstrates a strong capability in correctly predicting the "Fully Paid" category with a high number of true positives (TPs) at 3024. This suggests that the features associated with this class are well-captured by the model, allowing for accurate predictions. However, for "Charge Off", while there is a notable count of true positives at 415, there are also 259 instances that were actually "Charge Off" but were predicted as "Fully Paid". This misclassification indicates a challenge the model faces in distinguishing between these two outcomes. There are smaller numbers of misclassifications for other classes as well. For example, 73 instances of class 3 were misclassified as 'Fully Paid,' and there are 17 and 30 instances of classes 1 and 2, respectively, being incorrectly predicted as 'Charge Off.'

4. Discussion

4.1 Performance Comparison

In this section, we review the comparative performance of various machine learning models implemented

throughout this study. Each model was assessed based on its ability to accurately predict loan statuses, with a particular focus on identifying loans that would eventually be charged off, a critical factor for minimizing financial risk in lending practices.

The Naive Bayes model, despite its simplicity and quick execution, yielded an exceptionally low accuracy of only 2.6%, making it inadequate for our purposes. Its performance was particularly weak in predicting the 'Charge Off' status, with recall rates below 10%, indicating a fundamental limitation due to the violation of the conditional independence assumption within our feature set.

In contrast, the Logistic Regression model, especially its improved version employing the 'lbfgs' solver and class weight adjustments for handling imbalanced data, showed a promising accuracy of approximately 83%. It demonstrated a substantial ability to correctly predict 'Charge Off' cases with a recall of 84%, suggesting its efficacy for the bank's risk management.

The Support Vector Machine (SVM) with an RBF kernel also exhibited high accuracy at around 82% but struggled with a significant trade-off in its recall for 'Charge Off' statuses, recorded at 60%. This limitation underscores a potential risk of substantial financial losses, as a considerable proportion of risky loans might be erroneously approved.

The Random Forest model proved to be quite robust, offering good predictive accuracy and managing a 71% recall rate for 'Charge Off'. This model benefits from its ensemble approach, which improves generalization over the decision trees from which it is composed.

Finally, the Deep Learning model performed well in identifying 'Fully Paid' statuses but encountered challenges in distinguishing 'Charge Off' cases accurately. Although it captured a fair number of true positives, it also misclassified a significant number of such cases, indicating areas for model refinement.

4.2 Future Improvement

Adjust model to meet bank's goal. For example, with a target recall rate of 80% for charge off, which model can achieve the highest recall for fully paid?

To enhance the predictive performance of our models, particularly in improving the recall for 'Charge Off' predictions to meet the bank's strategic goals, several approaches can be considered. Firstly, further tuning of model parameters, especially for the SVM and Deep Learning models, could address the misclassification issues. Additionally, incorporating more granular features, potentially derived from enriched data sources or advanced feature engineering techniques, may provide deeper insights into the factors influencing loan defaults. Adopting techniques such as synthetic data generation to balance the classes in our training data could also help improve model learning conditions. Moreover, exploring alternative algorithms that might be more adept at handling imbalanced datasets or that offer different strengths in model interpretation could be beneficial.

4.3 Conclusion

This project has underscored the critical role of predictive analytics in financial risk management within the banking sector. By employing advanced machine learning techniques, we have been able to identify significant predictors of loan default, which can help refine the bank's loan approval processes. The insights gained from this study not only aid in risk mitigation but also enhance strategic decision-making, thereby contributing to the bank's operational effectiveness and financial stability. As we look to the future, continuing this line of research by experimenting with emerging machine learning techniques and expanding our data framework will likely yield further improvements in predictive accuracy. This ongoing refinement will ensure that the bank remains at the forefront of risk management technology, ready to adapt to an ever-evolving financial landscape.

Reference：

Team, Keras. “Keras: Deep Learning for Humans.” *Keras.io*, 2018, keras.io/. Accessed 24 Apr. 2024.

Kingma, Diederik P, and Jimmy Ba. “Adam: A Method for StochasticOptimization.” *ArXiv.org*, 2014, arxiv.org/abs/1412.6980. Accessed 24 Apr. 2024.

Geoffrey, et al. “Deep Learning with Parametric Lenses.” *ArXiv.org*, 2024, arxiv.org/abs/2404.00408. Accessed 24 Apr. 2024.