Using Machine Learning to Predict Bank Loan Defaults

Massimo Camuso

Introduction

With lending being a bank's primary way of making money, many banks are interested in knowing whether a borrower will default on their loan. Defaulting on a loan means lost money for the bank, and being able to predict if that borrower with default is valuable information. This project set out to use machine learning models to offer insight into the relationship between loan/borrower data and defaulting and predict if a loan will be defaulted. The models used were logistic regression, random forest, and XGBoost models on a bank loans dataset and compared each of their recalls and precisions to determine which model yielded the best results.

Approach - Preprocessing

As mentioned, this project aims to use a variety of machine learning models to predict if somebody will default on their loan or not given a loan/borrower's data. Before any of these models can be utilized, data preprocessing must first take place to ensure optimized usage of predictors. Luckily no missing values were found in the dataset, but since the dataset is so large (255,000+ rows of data) and since there are 16 predictors, preprocessing took some time. The dataset included 7 continuous features and 9 categorical features, and these features are as follows:

- Continuous features: age, income, credit score, debt-to-income ratio, loan amount, interest rate, and months employed
- Categorical features: number of credit lines, education level, employment type, marital status, has mortgage or not, has dependencies or not, has cosigner or not, loan term, and loan type.
- Target variable: default (binary)

The 7 continuous features were scaled with standard scalar to ensure that no feature dominated others in scale. When finding the best feature weights, gradient descent is susceptible to the scale of the data, so this step is essential. Some of the features, like DTI ratio, were extremely small in range (0.1-0.9) and needed to align with wider ranging features so that it would not be dominated in model weighting. However, other features had the opposite problem and had a range that was too big, like income and credit score. Because of the variety in ranges in the continuous variables, scaling all of them is the best approach so that no feature dominates in scale. It is also important to note that scaling is necessary for linear models such as logistic regression, but for random forest and XGBoost, scaling is not necessary. However, when fitting random forest and XGBoost, the data was still scaled.

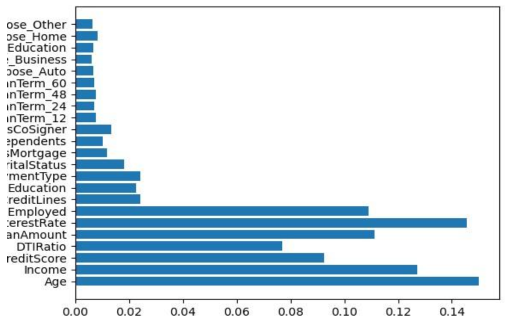
For the categorical features, all of them were changed to a numerical value so that they could be interpreted by the model. Education was changed using ordinal encoding to reflect that different degrees have different default rates (confirmed by graphing). The same was also done to EmploymentType because different employment types had differing ordinal default rates. MaritalStatus was also changed in the same way, because there was an ordinal relationship between marital status and default rates. For HasMortgage, HasDependents, and HasCoSigner, they were changed to a binary variable because the only values were yes or no, which were changed to 0 and 1, respectively. LoanPurpose was changed using one hot encoding since there was no ordinal relationship in the purpose of a loan and defaulting. Finally, LoanTerm was also changed with one-hot encoding even though it was originally viewed it as a continuous variable. The reason for using one-hot encoding was because these are fixed durations that represent the length of a loan, and the model could represent these durations as a number in the mathematical sense instead of the categorical sense. There is no mathematical continuity in these numbers, and with one-hot encoding we can compare if different loan terms have a relationship with higher defaults rates.

The final preprocessing task that was performed was building a correlation matrix to assess correlations and multicollinearity. Fortunately, there were no correlations in the dataset among features, meaning that multicollinearity was not an issue.

Approach – Modeling

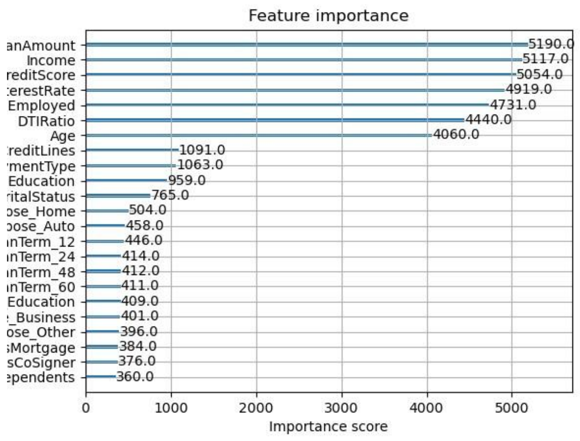
The first model that was trained and tested was a logistic regression model, which was the most tedious model to train. The reason for this is because the dataset is extremely imbalanced, with the ratio of defaults to non-defaults being 7:1. Unfortunately, logistic regression is not good at predicting heavily imbalanced data because it tends to favor the majority class, which in this case is non-defaults, and underpredict the minority class, which in this case is defaults. The models training performance metrics suffered because of this, so to combat this issue the data was resampled. Resampling is a technique where the minority class is artificially upscaled to match the majority class and allow models to train on balanced data to interpret which features are most significant in predicting the target variable. After resampling, the training metrics significantly improved, yielding both precision and recalls above 70%. The performance of the model did not significantly differ among LASSO, Ridge, and Elastic Net models, so LASSO was chosen to keep it simple. The most significant predictors in the model were interest rate, loan amount, age, having a cosigner, credit score, income, months employed, and having a mortgage, which are a mix of continuous and categorical variables. Another aspect of testing was adjusting the threshold for which the model predicts a default. The default probability of 50% is inadequate because of the imbalance, but we found that the threshold 0.6 yielded the best results. Another special result of the logistic regression model was that it achieved the highest AUC score compared to the random forest and XGBoost models, which will be discussed in the results section.

The next model that was trained and tested was random forest. The reason random forest was used is because it may better handle the data imbalance (no resampling required) and could capture nonlinear relationships between the dataset's features and the target variable. Before fitting the model, random forests feature importance graph was used to analyze which features may contribute the most to the models' predictions. This was the yielded graph:



According to random forest, the continuous features are significantly more important at predicting defaults than the categorical features. After using GridSearchCV to train the model and tweaking the parameters, it was found that the best parameters were n_estimators = 500, max_depth = 15, min_samples_split = 2, min_samples_leaf = 3, max_features = 0.8, and class_weight = {0:1, 1:7} (manual class weighting to account for the imbalance). F1.5 score was also used to find the optimal probability threshold for predicting a default, which is a modification of the classic F1 score that slightly favors recall over precision. Finally, the predicted probabilities of the model were calibrated with the CalibratedClassifierCV to further refine predictive performance. The results of random forest will be discussed in the results section of this report.

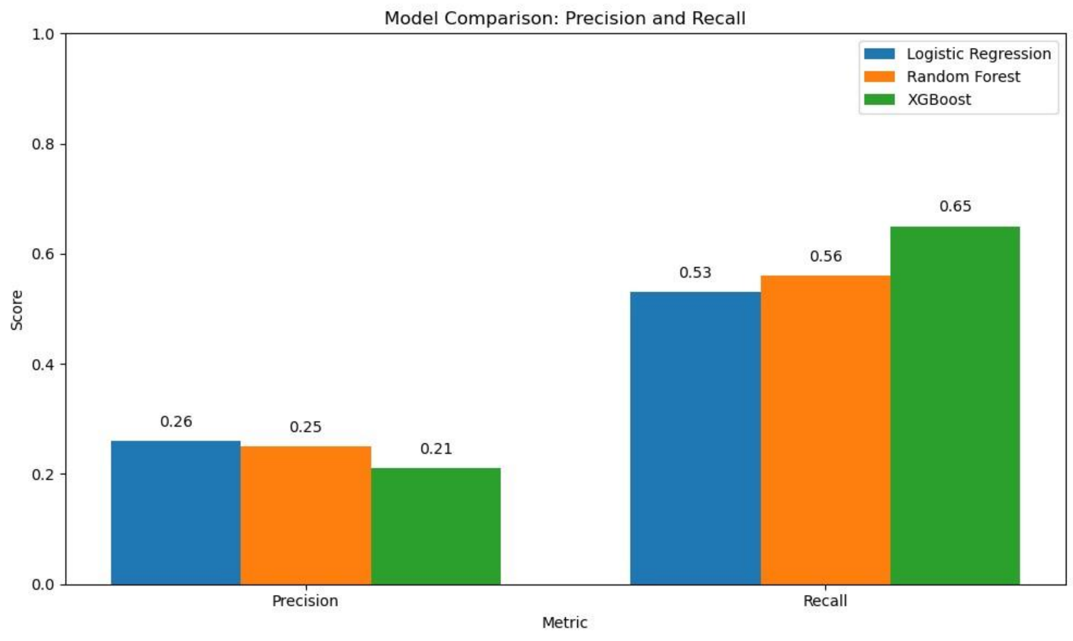
Finally, the last model that was trained and tested was XGBoost. XGBoost is similar to random forest in the sense that it better handles imbalance (no resampling needed) than logistic regression and is a nonlinear model. After testing random forest, XGBoost was evaluated to see if it could yield any better performance metrics. Also similar to random forest, XGBoost has a feature importance graph to see which features were driving predictions the most. Here is the graph:



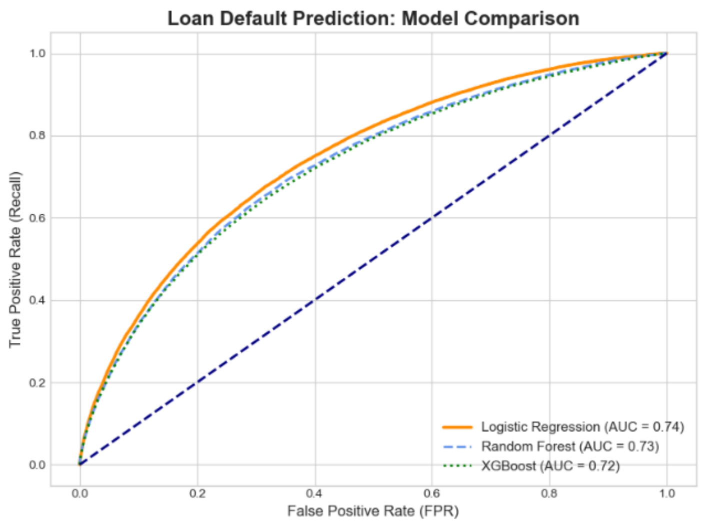
Plotting XGBoost's feature importance graph confirmed that the features were important, as the features that XGBoost found important were all the continuous features which matched random forests importance graph. After using GridSearchCV to train the model and tweaking the parameters, it was found that the best parameters were scale_pos_weight = 13 (account for data imbalance), eval_metric = 'aucpr', colsample_bytree = 1.0, learning_rate = 0.1, max_depth = 10, min_child_weight = 1, subsample = 0.8, n_estimators = 200, reg_alpha = 5, reg_lambda = 5 (strength of regularization). Similarly to random forest, the best threshold was found by optimizing the F1.5 score, which was 0.118, and calibrated probabilities with CalibratedClassifierCV to further optimize performance. Now the results of the models will be discussed.

Results

Overall, the performance of each model on predicting the testing set was very similar. Here is a graph describing the precision and recall for each model:



The differences in precision among all three models are quite marginal, with logistic regression having the best precision at 26%. This means that roughly 25% of the loans flagged as 'high-risk' (loans predicted by the model that are above the probability threshold) will actually default, and that 75% of the loans flagged as 'high-risk' are false alarms. Among all three models, it was difficult to exceed a precision of around 20% without recall being severely penalized, and the highest precision that was achieved (if recall is severely penalized) was only about 50%, showing that all models struggled to achieve a high precision with a 7:1 imbalance. As for recall, the differences among all three models are also marginal, although they differ greater than precision. XGBoost is the clear winner among the three, achieving a recall of 65%. This means that the XGBoost model catches 65% of actual loan defaults but misses 35%. These results are decent and are better than random guessing, but again, the 7:1 data imbalance makes it difficult to improve these metrics. As mentioned earlier, logistic regression was able to achieve the highest AUC-score of 74%, with XGBoost and random forest trailing slightly behind with an AUC-score of 72%. This result shows that more complex models are not always going to perform better than simpler ones. Even though random forest and XGBoost tend to capture more complex patterns in data being tree based models, logistic regression was slightly more effective at distinguishing between defaults and non-defaults despite being the simpler model. Here is a graph describing each models ROC curve:



It is worth discussing why in each models recall exceeds precision. Because this is a banking loan dataset to assist a bank, recall should be prioritized. Since recall measures for what percentage of defaults are actually caught and missing a default can be very costly for a bank, prioritizing recall is important in order to save as much money as possible. On the other hand, precision measures the percentage of loans that are flagged as ‘high-risk’ that will actually default, which is less costly for a bank. If a person is flagged as a default but does not actually default on their loan, the cost of that missed customer is much less than the cost of a customer actually defaulting for a bank. Thus, because of the nature of these metrics and the problem that is trying to be solved, prioritizing recall was sensible. Depending on the probability threshold that we set, recall and precision could increase or decrease. A higher recall could have been achieved at the expense of lowering precision, but a probability threshold that bumped recall higher than precision while minimizing the discrepancy between the two was prioritized, which is why the threshold was chosen based off of what maximized the F1.5 score, which prioritized recall. The largest discrepancy between recall and precision is found with XGBoost, but XGBoost’s precision performance was only 4% lower than the next best, which was random forest.

Conclusion

In conclusion, knowing if a person will default on their loan is crucial information to a bank, and defaults can lead to costly losses in profits. Machine learning models are a great way to help banks make informed decisions and understand what factors are most significant in predicting if a person will default on their loan. After preprocessing, it was found that the dataset's continuous features and select categorical features were significant in the variability of loan defaults. However, due to the imbalanced nature of loan default data, it makes overfitting a real issue. Defaulting is a naturally rarer occurring event than not defaulting, and while resampling allows the model to understand which features drive defaults, it falsely teaches the model to expect balanced data which is rarely ever the case. Thankfully, models like random forest and XGBoost do not require being trained on balanced data, but that still does not change the fact that prediction performance with these models still suffers when data is heavily imbalanced. Regardless, after training, calibrating probabilities, and finding the best probability threshold, a peak precision of 0.26 and a peak recall of 0.65 was achieved. These metrics can be increased or decreased depending on the probability threshold of the given model, and these metrics show that machine learning is a valuable tool that can advise a bank on what decisions to make. If this project was to be done again in the future, perhaps other models could be tested to see if they perform better on imbalanced data and yield better prediction results. Regardless, this project taught me a lot about the machine learning process and the types of issues that arise. It was satisfying to apply the concepts we learned in class to a real-world scenario, and I believe this project has better prepared me for the challenges I will face in a real-world scenario.

Acknowledgements

- Kaggle – Source of data
- StackExchange – overall great source for understanding topics and getting answers to your questions.
- DeepSeek AI – helped with debugging

Thank you for reading my paper!