Predicting Loan Defaults for Lending Club

Ernest Stephenson, Li Li, Sibi Rajendran
Team 6

Introduction

Lending Club is a United States peer-to-peer lending company that was founded in 2006. It is the first peer-to-peer lender to trade publicly and provide a secondary market for loan trading. It currently trades on the New York Stock Exchange and has a market cap of $2.34 billion. It is the world’s largest peer-to-peer lending company with almost $25 billion in loans being originated through its platform [1].

Lenders don’t lend directly to borrowers, as is the case with some other peer-to-peer lenders. Instead, borrowers apply for loans online and Lending Club reviews the loan application based on the borrower’s credit score, credit history, debt-to-income ratio, and other factors. From there, Lending Club rejects approximately 90% of the loan applications as the company has decided to only focus on high creditworthy borrowers. The loans can be from $500 to $40,000 for a 3- or 5-year term and if accepted, they are then placed on the Lending Club website for investors to browse. Investors then can choose to buy “notes” from Lending Club that are graded A to G according to the risk of default. The money that the investors provide is the money that the borrowers receive for their loan amount. These loans provide lower rates than a typical bank because of Lending Club’s low-cost structure with operations fully online and no branch infrastructure. Lending Club makes money off of these loans by charging borrowers an origination fee and charging lenders a service fee.

The goal of this project is to predict loan defaults from the Lending Club database. This will help Lending Club in their initial decision of whether to grant borrowers loans or not. Our prediction doesn’t take the time of default into account at all, but just predicts if a loan will default at any time over the term of the loan. We downloaded our dataset from the Lending Club website as a CSV and used all available loan data from 2007 to 2011. Lending Club has data available for later years but we decided to only use loans that aren’t still being paid on. Since

Lending Club only provides three- and five-year loans, all loans that originated in 2011 or sooner would be defaulted on or fully paid off by now. The original data set that we used had 42,538 observations and 110 predictors.

Data Cleaning

After downloading the data, we focused on reducing the number of predictors. Our first step in doing this was by deleting all columns with a large amount of N/A values. This data set had a lot of missing data and we removed 58 predictors by deleting all columns where more than 80% of the values were N/A's. Then, we reduced the number of predictors from 52 to 18 by manually going through and deleting all columns that we thought wouldn't have any predictive value. This included deleting a lot of variables that had to do with the investor's information. We didn't think that knowledge of the investor's finances would help in prediction at all since they have no participation in paying back the loan. We also deleted predictors such as application type, number of tax liens, and number of bankruptcies, where nearly all of the loans had the same value. Since there was no differentiation in these predictors between loans, we didn't believe that they would provide any predictive value. Finally, we deleted predictors that didn't make any conceptual sense to us in predicting loan status such as zip code, state, and loan description. After looking at the data and the consistency of loan default rates among states (with the exception of a few outliers in very small sample size states), we decided that geography would not be helpful in prediction. The loan description column was filled with long descriptions where some borrowers rambled on about why they needed the loan and mentioned many unnecessary details. We decided there would be no way to quantify this column and that it wouldn't be helpful.

Our final data set we used for our model had 42,538 observations and 18 predictors. All 18 of the final predictors are listed in the table below with their descriptions. We created a binary response variable called "is_bad" that determined whether each loan was a default or not. There were originally six classes for this variable, but we assigned a zero value to the fully paid loans and assigned a value of one to the other five classes. Those five classes were Charged off, Does not meet credit policy: charged off, Does not meet credit policy: fully paid, Late 16-30 days, and Late 31-120 days. We decided to group the classes like that based on previous work [2][3] that grouped the binary variable that way and because we wanted to separate fully paid loans from loans that would cause Lending Club any type of trouble. Next, feature engineering is a very important part of machine learning and we worked to notice relationships between variables and attempted to create new variables that would have predictive value. The first new feature we created is called “time_since_first_credit” and is the “earliest_cr_line” minus the “issue_d”. This ended up being one of our variables with the most predictive value, as you will see later in the paper. Next, we created a feature called “perc_recv” which is the amount of principal received (one of the 52 trimmed predictors) divided by the “loan_amnt”. This ended up having predictive value, but the problem is that this information wouldn’t be available to Lending Club at initiation when borrowers are applying for loans. Therefore, we didn’t use this feature in our model but it could potentially be used by Lending Club in another model that is used to predict borrower’s default risk over time as they pay their loan.

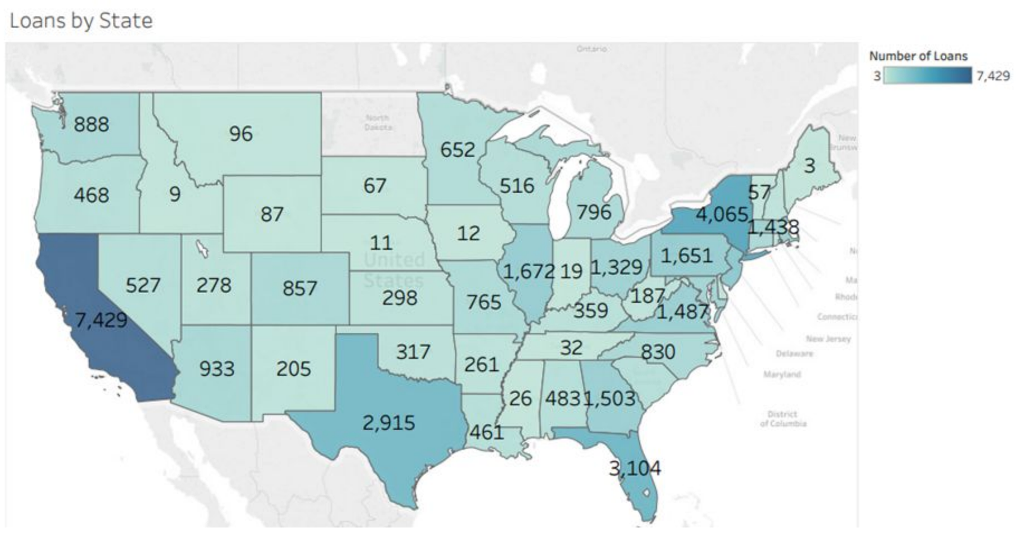
Predictor	**Description**
loan_amnt	Listed amount of the loan applied for by the borrower
term	The number of monthly payments on the loan (36 or 60)
int_rate	Interest rate on the loan
installment	The monthly payment owed by the borrower
grade	Lending Club assigned loan grade
sub_grade	Lending Club assigned loan subgrade
emp_length	Employment length in years (0-10+)
home_ownership	The home ownership status of the borrower (Rent, Own, Mortgage, Other)
annual_inc	The self-reported annual income of the borrower

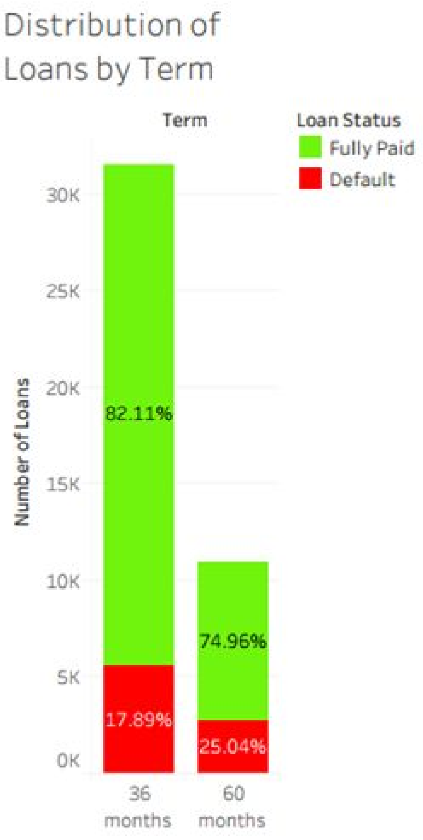
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
issue_d	The date which the loan was funded
dti	A ratio calculated using the borrower's total monthly debt payments divided by the borrower's self-reported monthly income
earliest_cr_line	The date the borrower's earliest credit line was was opened
open_acc	Number of open credit lines in the borrower's credit file
revol_bal	Total credit revolving balance
revol_util	The amount of credit the borrower is using relative to all available revolving credit
total_acc	The total number of credit lines currently in the borrower's credit file
time_since_first_credit	earliest_cr_line minus issue_d

Before we began working on our model, we looked for previous work done on this data. There were multiple projects that used logistic regression, but they all had problems with low accuracy and a high number of false negatives. We decided that false negatives would be more harmful for Lending Club than false positives, so we focused on reducing the amount of those. Incorrectly predicting a bad loan as a good loan (false negative) is more costly because it leads to the direct loss of money by having a borrower default. As opposed to incorrectly predicting a good loan as a bad loan (false positive), which would just be an opportunity cost of turning away profitable customers. This isn't a good thing but is easily fixable by just generating more business and doesn't have as much of a long-term negative effect as false negatives. We also looked into K-nearest neighbors (KNN) but did not find any projects that used KNN on this data set. This is because the data set is too large and computation time would be an issue, therefore we also decided not to use the KNN method.

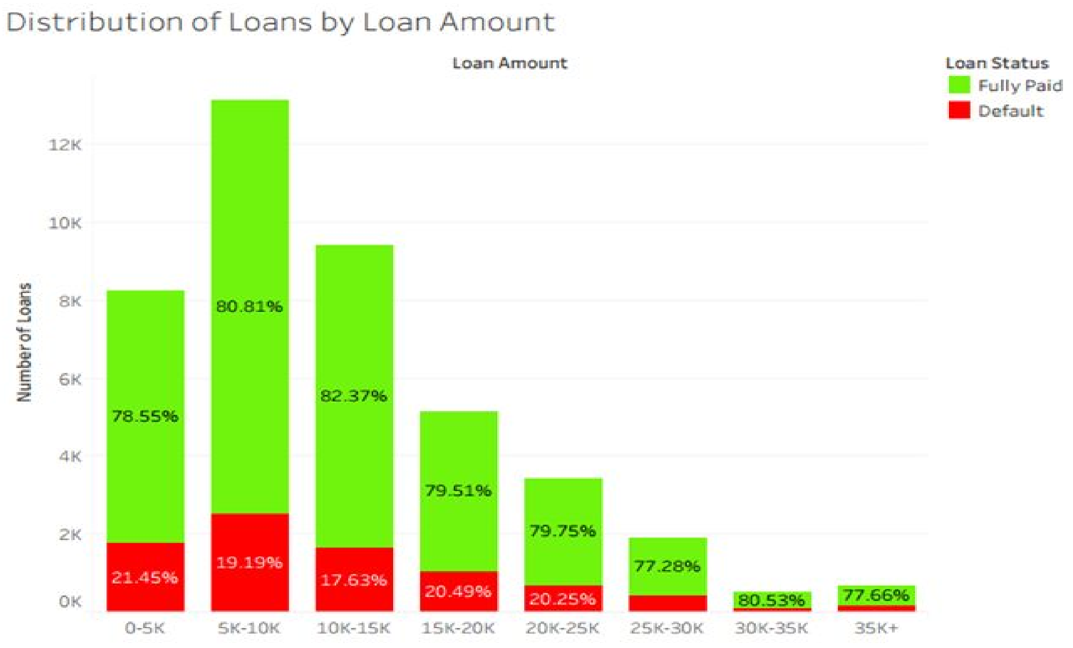
Exploratory Data Analysis

Below are a few visualizations and graphs that we looked at as we explored the Lending Club data. Directly below you can see the amount of accepted loan applications by state. As expected, the states with the most amount of loans are the most populous states (California, New York, Florida, Texas).

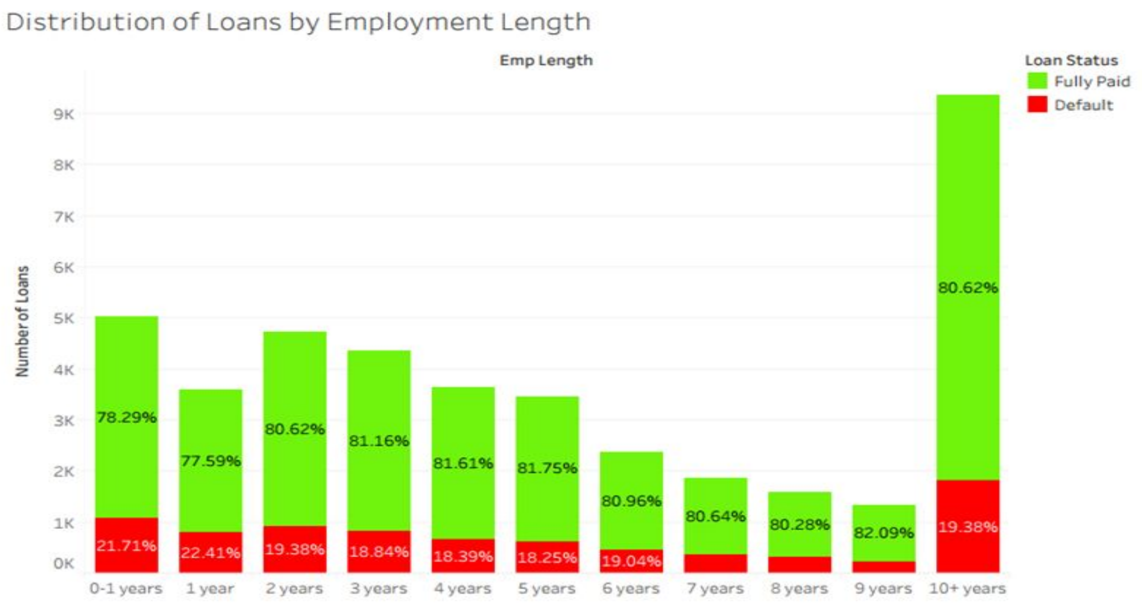




To the left, you can see the distribution of loans by term. Approximately three-quarters of the loans are 36-month loans, which is good news since those have a higher percentage of fully paid loans. Below this you can see the distribution of loans by loan amount. Even though loans on Lending Club are available for anywhere from $500 to $40,000, more than 70% of the loans made are from $500 to $15,000.

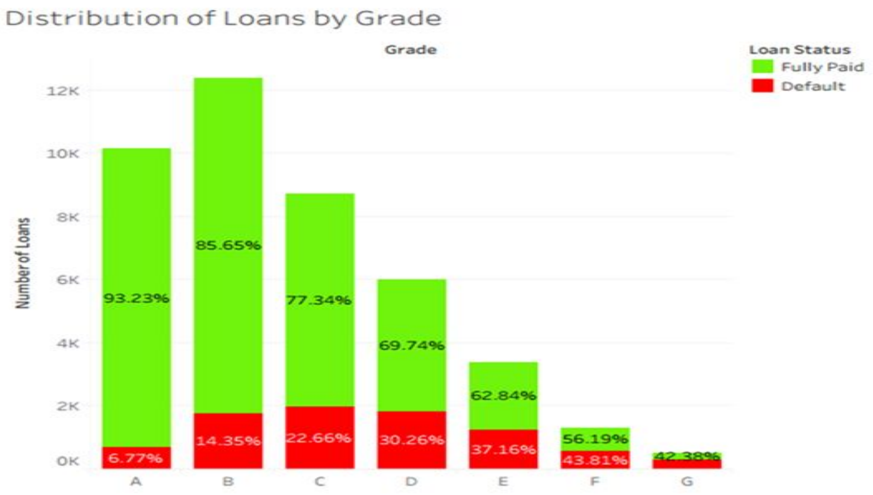


In the graph below is the distribution of loans by employment length. Lending Club purely focuses on borrowers with high creditworthiness. By far, the largest amount of loans are given to borrowers with employments lengths of 10+ years, which would indicate a more predictable borrower and fall in line with Lending Club's policy.

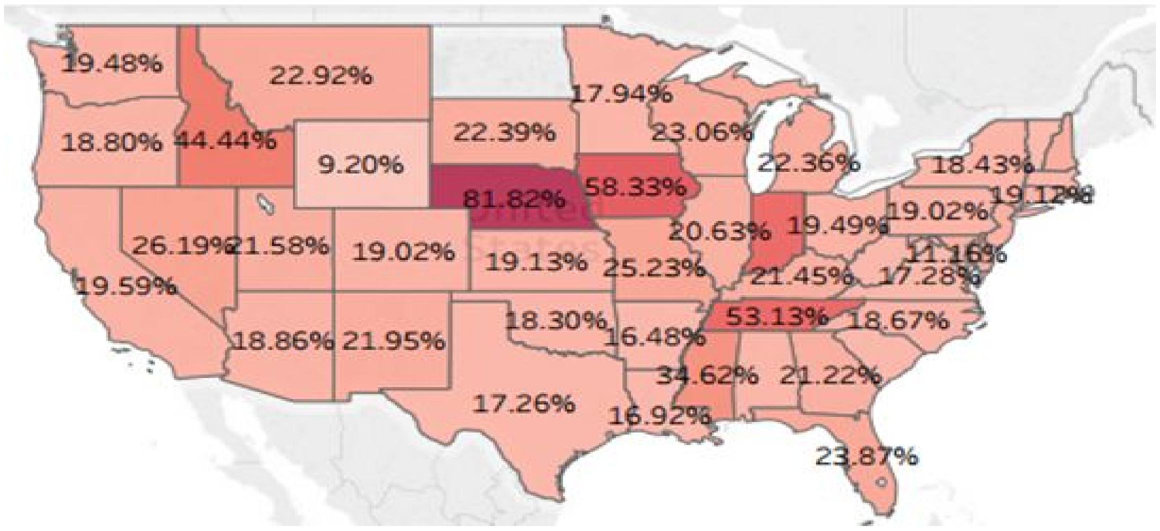


Lending Club assigns a loan grade to each loan for investors to know the quality of each loan with A being the highest quality and G being the lowest. Below you can see the distribution of loans by these given grades. Default rates obviously get higher as the loan grades get lower.

But staying in line with their mission of focusing on high creditworthy borrowers, you can see that approximately three-quarters of the loans are given to borrowers that are assigned loan grades of A, B, and C.



Finally, we have a map of the percentage of loans in each state that are defaults. Across most states, there is a very reasonable default rate between seventeen and twenty percent, but there are a few outliers. The states of Idaho, Nebraska, Indiana, and Tennessee have abnormally high default rates but there are an extremely small amount of loans in each of these states so they are just random outliers and we didn't adjust for this in our model at all.



Modelling

Training and Test Splits

The dataset containing 42,538 observations, as a convention, was split as training (70%) and testing (30%). Most of the hyperparameters available for the machine learning algorithm were tuned on the training set using tune.grid and cross-validation. Even in cross-validation, the number of folds were kept as a tunable parameter. The objective was to strike a balance in increasing both the accuracy of the model and decreasing the number of false negatives of the model.

Machine Learning

Logistic regression and SVMs have been tried before as a part of modeling <insert citations>. Also, these results tend to vary a lot due to the fundamental difference in choosing predictors for the problem. Our goal was to let LC get a fair idea of whether a borrower would default or not and hence, the predictors chosen were the ones that are usually available to LC before the loan period begins. This limits the choice of predictors. Prior attempts in this field have chosen other predictors which include details about the funds investors have promised to invest but these are available only after the investors are chosen and the loan period has begun. For this reason, we have not included these predictors in our analysis.

The prevalent failure of the aforementioned methods and promising results in trees influenced us to dwell more on tree based methods. The major methods using trees are decision trees, random forests and the relatively recent XGBoost methods.

Decision Trees

Package Used : rpart

Using default parameters of the tree gave an accuracy of 81.5% but this was at the cost of recall (0.13) and good precision (0.98). This problem is one that's seen in all prior work. On an average, most models end up predicting 75% of default loans as good loans.

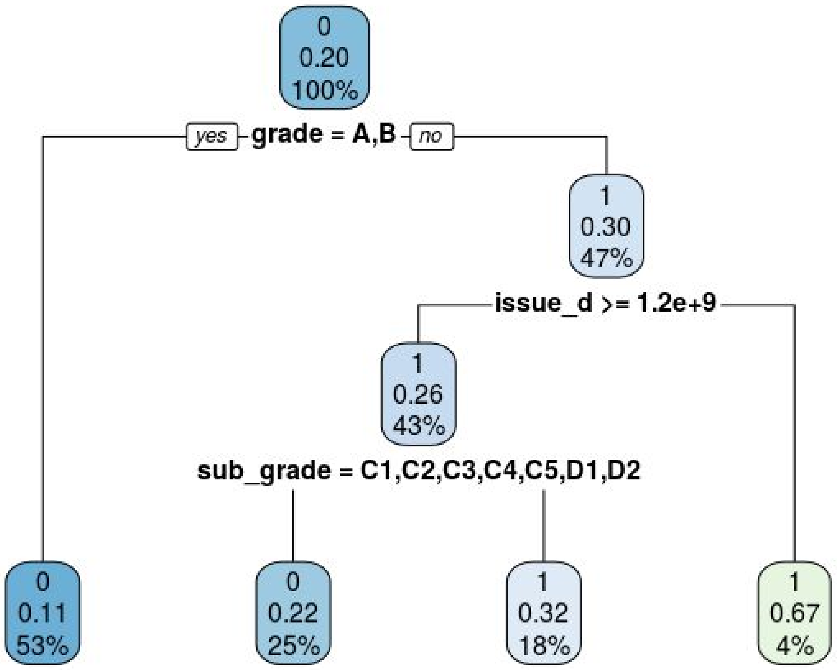
We use *rpart* package in R - it uses several methods described in the book, CART (Classification And Regression Trees). A key reason why we chose *rpart* was because it has inbuilt ways to handle missing values; manual imputations are thus deemed redundant.

To give negative classes more penalty when misclassified, we tune the parameter 'scale_pos' and decide on a 1:3 scaling ratio for the underlying outcome variables. Also, instead of allowing the default threshold of 0.5, we get the outcome variables as the probabilities of a test data point belonging to one of the two classes and tune the threshold so as to give us fewer false negatives.

We notice that the important predictors using rpart tend to be *grade, interest rate, issue date, revolving balance.* After tuning all the parameters, we get the best possible results.

	Accuracy	Recall
Before Tuning	0.81	0.13
After Tuning	0.66	0.60

Final Model



Random Forests

Package Used : randomForest

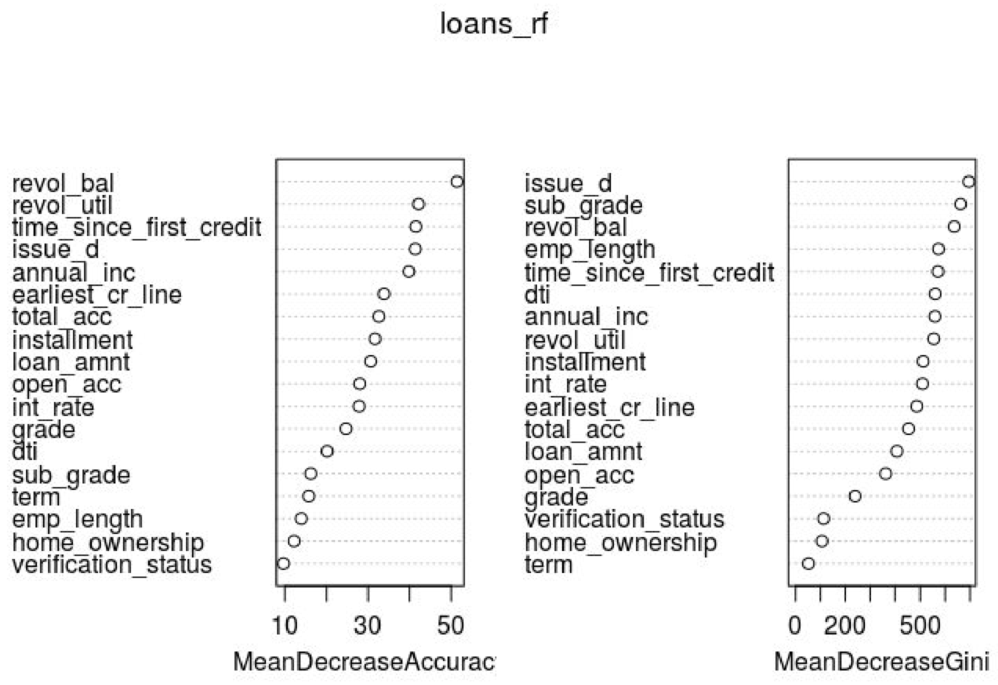
In theory, Random Forests should perform better than decision trees as they reduce variance by building n-trees. It essentially builds n decision trees using a bagged sample of data from the training set and using the rest of the data set for validation.

As a thumb rule, we start by setting mtry (number of predictors) to be the square root of the number of columns in the dataset. Also, we tune the number of trees parameter from 1 to 50. After plotting a graph between error and number of trees (500), we notice that after the error converses well within 50 trees and hence, there is no point in training using 500 trees. Similar to the previous analysis, we predict the probabilities for each observation belonging to one of the two classes and tune the threshold to decrease the false negative rate.

By default, similar to the previous analysis, the default parameters give us an accuracy of 82% (0.8125 to 0.8255 under a 95% confidence interval) and a recall of 0.08. We tune and increase the recall at the cost of accuracy. We get an accuracy of 68% but we have increases the recall to 0.70.

	Accuracy	Recall
Before Tuning	0.82	0.08
After Tuning	0.68	0.70

We also check the variable importance plot.



Inference : revol_bal, revol_util, time_since_first_credit, annual income are important predictors as seen from the plot. According to the gini index, issue_d, sub_grade, revol_bal, emp_length are better predictors in terms of separating the two classes.

XGBoost

Packages Used : xgboost, data.table, plyr

eXtreme Grading Boosting (XGBoost) is an open source library which provides a tree based gradient boosting framework. It was first used in Higgs Machine Learning challenge and was developed by Tianqi Chen as a research project. It won the competition and has since been used in several machine learning and data science competitions.

It is a supervised learning algorithm which uses an objective function <insert objective function> which contains a training loss term and a regularization term. Subsequently, by building decision stumps and updating the objective function by utilizing the errors, it optimizes the objective function on the training set. By specifying parameters such as the number of folds in cross-validation, maximum depth to which the trees should be built, the learning rate etc., we can tune the xgb model. One major constraint in operating with xgboost is that it uses a data.table/matrix in all its calculations. It becomes imperative that we switch our datasets from a data frame (containing different types of variables) to a numerical matrix.

As with previous training models, we first let the default parameters output train a model and then we tune it using a grid search. The parameters that can be tune are number of folds in the cross validation, number of trees (or rounds), maximum depth to which the trees can go, learning rate, scaling ratio etc. The results are shown below.

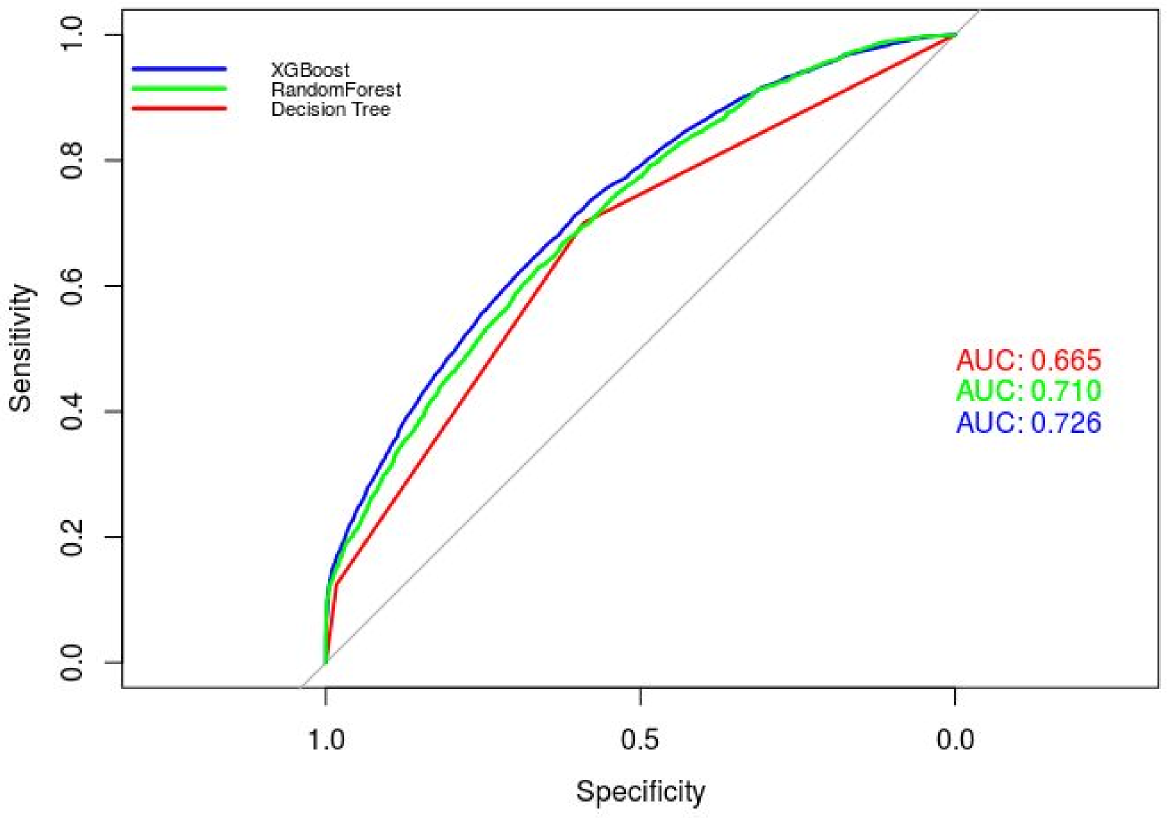
	Accuracy	Recall
Before Tuning	0.68	0.34
After Tuning	0.74	0.80

The parameters in the final xgbmodel are :

nrounds.cv	100
nfolds	4
scale_pos_weight	0.4
max_depth	5
eval_metric	“auc”
max_delta_step	1

Comparison

Since we do not have an estimate of how much loss is incurred due to delinquent loans, we cannot say for sure at what threshold LC will begin making profit. To get around this problem, we plot the RoC curves of each of the model and decide which one might be a better fit. It would finally be up to the user of the algorithm to decide which model to go with. The XGB model with a low false negative rate and comparatively high accuracy would be the recommended best model for this analysis.



Comparing the three tree-based models

Algorithm	Accuracy	Recall
Decision Tree	0.66	0.60
Random Forest	0.68	0.70
XGBoost	**0.74**	**0.80**

We decide to use the XGBoost model since it has a much higher accuracy and lower false negative rate as compared to other models.

Strengths:

- The computation time of trees is much less than KNN, which takes us too much time to process.
- Trees are more flexible and more scalable to large datasets than any other models we tried to use. We have 42,538 data points in the original data set. But if we include more, we need a model which could scale or accommodate rapid increase in size quickly and easily. So we use models which are scalable.
- The XGBoost model is fast and we can customize parameters in this model.

Weaknesses:

- XGBoost model is not easy to interpret. Unlike linear regression where we can see the coefficient of each predictor clearly, in our model, the direct relation between predictors and response variable cannot be determined.
- Also for XGBoost model, it is difficult to tune because of the large number of parameters.

Future Scope

We would like to improve our model for prediction and there are many ways to do that.

- Include FICO scores in the dataset and analysis. FICO is a person's credit score which can measure the credit risk of clients accurately. In our database, we don't have this variable but if we had it, we can make a more precise prediction.
- Include data from subsequent years. Our data is from 2007-2011 and this is the only available complete payment cycle on the Lending Club website. We are willing to include more data as time passes to make predictions using a larger data set.
- More feature engineering could be used in prediction. For the limitation of predictors

and knowledge, those three tree based models are relatively reasonable, but actually, the prediction model could be more complicated and detailed in real life.

- After modeling, we came up with a really fresh idea to make an application for Lending Club and people only need to input parameters like clients' annual income, debt-to-income ratio, etc. then the default probability pops up automatically.

Conclusion

Tree based models have relatively better accuracy as compared to other models and that's why we see them as better models. When Lending Club uses the model, we want to assign each loan a probability for Lending Club to make their own decision, not to just automatically accept or deny loans. Also, Lending Club can set the threshold in the model according to different conditions, which would help Lending Club avoid accepting applications from customers who have high default risk with efficiency and flexibility.

References

[1] Statistics from https://www.lendingclub.com/info/statistics.action

[2] http://cs229.stanford.edu/proj2015/199_report.pdf

[3] https://rpubs.com/torourke97/190551