UNIVERSITY OF ILLINOIS CHICAGO

Assignment 1 (A & B)

IDS 572 – DATA MINING

ZOHAIB SHEIKH, SHUBHAM KHODE & ANIRUDHA BALKRISHNA

Lending Club Business Model (Q1)

What is Peer-to-Peer (P2P) / Marketplace / Alternative lending?

In the traditional process of borrowing and lending, banks have always acted as intermediaries and contributed to overall increase in cost of transaction since they make significant profit off the spread. But today technology and innovation are making possible a new generation of financial services that are more affordable and more available. As the process becomes online, it possible for a third party to directly match idle borrower with its lender/investor, all without the involvement of retail banks or credit card companies. The idea is like what Airbnb and Uber are doing to traditional lodging and transportation industries – marketplace lending model cuts several links out of today's banking chain and translates that savings into low interest loans.

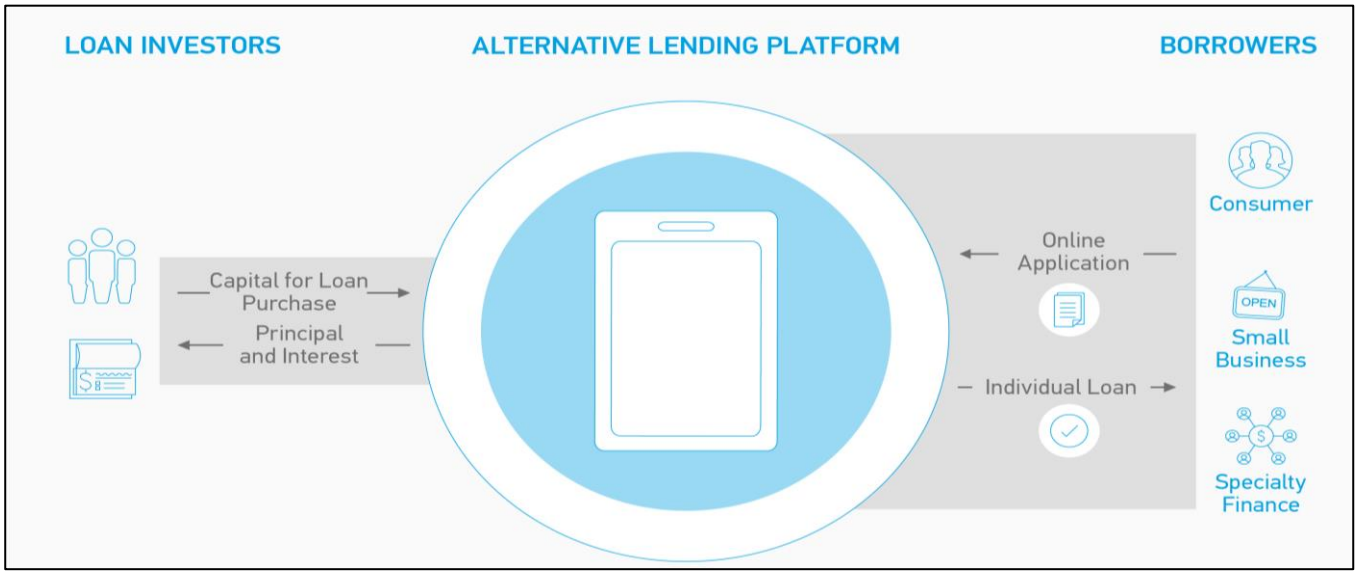


Figure 1. P2P model

Business Model

The key stakeholders in P2P lending are the loan investors/lenders and the borrowers. The P2P platform role is of an intermediary which supports the whole process and all stakeholders. As an intermediary, the platform scores borrower quality through their data-driven models and enables lenders to make good decisions. Owing to low operational costs, they reduce transaction costs for both borrowers and lenders, leading investors to achieve higher returns while borrowers receive lower interest rates.

How P2P platform make money?

The P2P platform charges both origination fees and ongoing loan service fees. By providing additional value to borrowers and lenders, they maximize the number of transactions happening on their platform and further propel the growth of their business.

Advantages of P2P lending for borrower

- *Low interest rates and easy loan repayment with fixed scheme over a set period.*
- *Ability to get unsecured personal loan (without putting any collateral), creditworthiness evaluation based on data points aside from FICO scores.*
- *Borrower convenience owing to digitization, transparency, and minimal regulatory constraints.*

Advantages of P2P lending for lender/investor

- *Predictable, stable, and higher yields at relatively low, flexible durations due to amortizing structure of alternative loans.*
- *Marketplace loans allow for diversification of portfolio by creating a great asset class.*

Advantages of Lending Club

- *Superior Credit Scoring model*

P2P lending platforms have a competitive advantage over traditional banks since they incorporate additional data sources and parameters to improve upon FICO credit score. They use their own proprietary data as feedback into the model, which in turn improves model's performance and accuracy. Better accuracy results in lower rates to borrowers, and then more borrowers flock to the platform, driving more data into the model.

In Figure 2, the 45-degree line is the FICO-score-only model while the bowed lines represent the P2P lending models that incorporate other sources of data with FICO. The AUC represent performance jump by accepting more good borrowers and rejecting more bad borrowers than FICO alone.

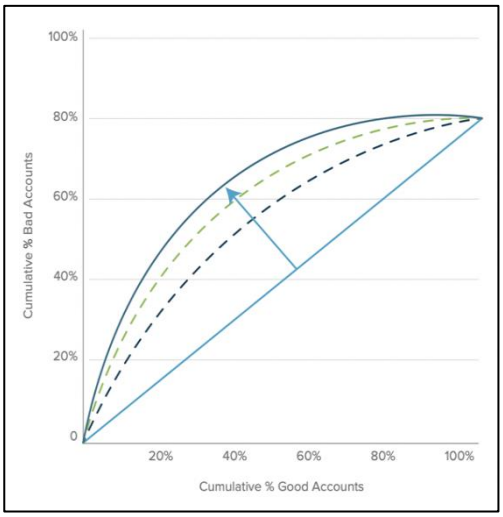


Figure 2. FICO vs P2P model

- *Full spectrum fintech marketplace bank*

The acquisition of digital bank "Radius" by Lending Club made it the only full spectrum fintech marketplace bank in the U.S. This move will grow their member base through a broader set of product offerings, including deposits, that will also leverage the strength of LendingClub's technology platform and marketing capabilities. Additionally, it will deepen the customer relationships and generate more data to add to its market-leading data sets and continue to refine the bank's underwriting algorithms.

SOURCES

1) https://www.morganstanley.com/im/en-us/financial-advisor/insights/investment-insights/an-introduction-to-alternative-lending.html
2) https://foundationcapital.com/wp-content/uploads/2020/04/FC_CharlesMoldow_TrillionDollarMarket.pdf
3) https://www.prnewswire.com/news-releases/lendingclub-receives-regulatory-approvals-to-acquire-radius-bancorp-301210498.html

4) https://digital.hbs.edu/platform-digit/submission/lending-club-creating-the-marketplace-lending-business-model/
5) https://www.financereads.in/p2p-lending-risks-and-rewards-for-stakeholders/
6) https://www.lendingclub.com/company/about-us
7) https://www.alliedmarketresearch.com/peer-to-peer-lending-market

Data exploration (Q2)

1. **What is the proportion of defaults ('charged off' vs 'fully paid' loans) in the data? How does default rate vary with loan grade? Does it vary with sub-grade? And is this what you would expect, and why?**

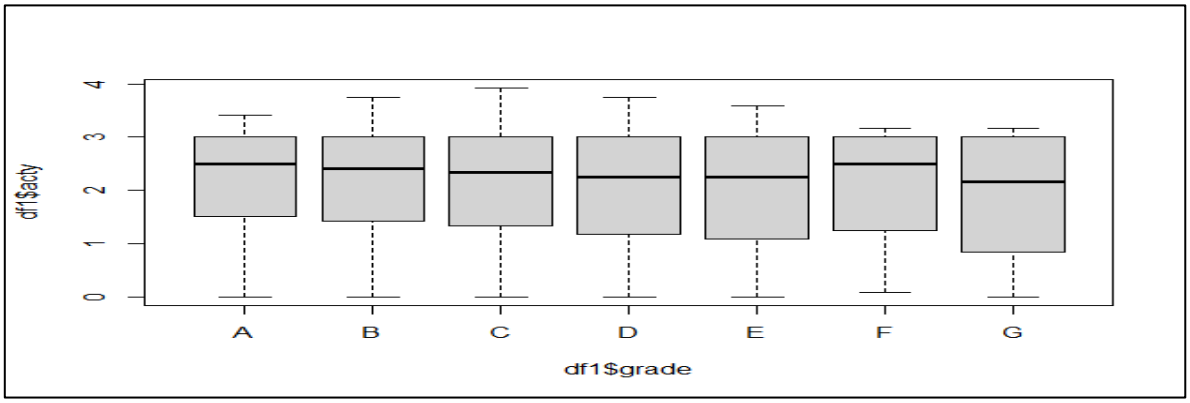
Out of total loans sanctioned, 13.8% of them have defaulted. Default rate varies greatly with the loan grade as expected. The lower the loan grade, more is the default rate. Grade C loans have default rate of 18% as compared to 5% of grade A loans. As the quality of loan applications decrease the higher the chances of being defaulted which is evident in the data.

2. **How many loans are there in each grade? And do loan amounts vary by grade? Does interest rate for loans vary with grade, subgrade? Look at the average, standard deviation, min and max of interest rate by grade and subgrade. Is this what you expect, and why?**

Higher grade loans such as A, B and C constitutes 81% of total loans sanctioned with B assigned to 33% of total loans, A to 22%, C to 26%. Similarly, the loan amount varies with grade, higher grades having the highest total loan amount and smaller total amount for lower grade loans. Similarly, as expected, the interest rate also varies with the quality of loans, with higher grade having lower rates as compared to lower grades. Grade A loans are offered at an average of 6% interest rate against 27% for G grade loans. The same follows for subgrade within each loan grade. Higher the subgrade, lower the interest rate. Loans with subgrade B1 are offered at an average interest rate of 9% as against 12% for B5. A similar pattern is seen for the spread of interest rate.

3. **For loans which are fully paid back, how does the time-to-full-payoff vary? For this, calculate the 'actual term' (issue-date to last-payment-date) for all loans. How does this actual-term vary by loan grade (a box-plot ca n help visualize this)?**

The average time in years to fully payoff the loan is 2.6years for grade A. There isn't a particular pattern observed between the average payoff time vs grade. This is evident due to the fact that the loan term for all sanctioned loans is 3 years.

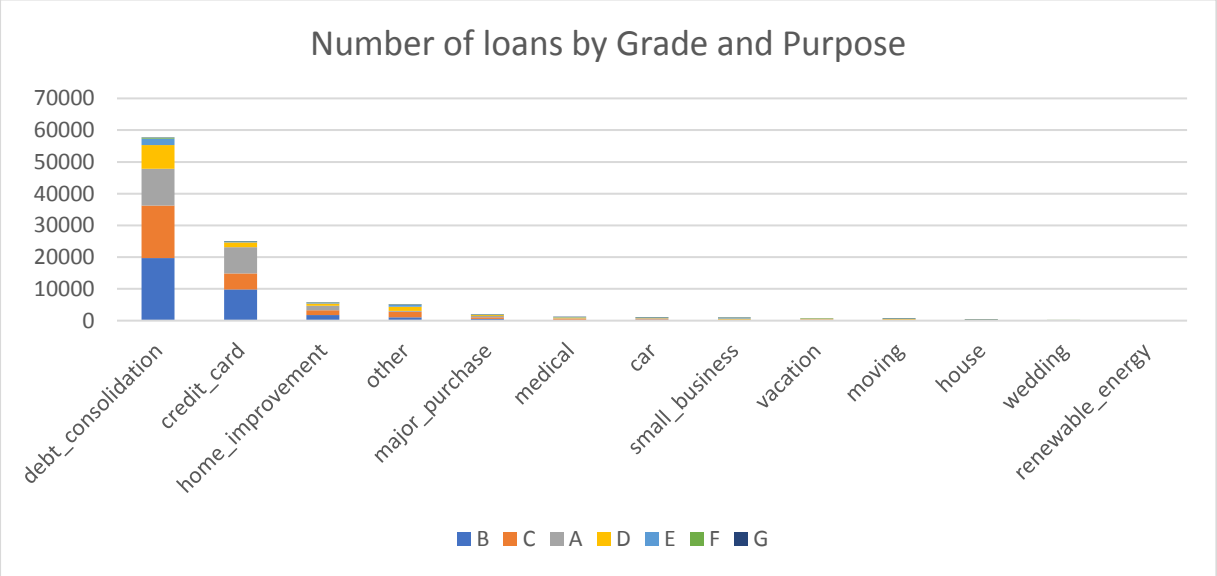


4. **Calculate the annual return. Show how you calculate the percentage annual return. Is there any return from loans which are 'charged off'? Explain. How does return from charged -off loans vary by loan grade? Compare the average return values with the average interest rate on loans – do you notice any differences, and how do you explain this? How do returns vary by grade, and by sub-grade. If you wanted to invest in loans based on this data exploration, which loans would you invest in?**

Percentage annual return can be calculated by averaging the monthly return and then extrapolating it to 12 months. Charged-off loans have negative average annual returns which varies with loan grade. Higher grade charged-off loans have lower average annual returns as grade G loans have -14% average annual return as compared to -11% for grade A loans. Some charged-off loans have positive annual returns which is driven by the total interest payment.
We observe that the interest rate increases as we move to lower grade loans. However, the average return values decreases as we move to lower grade loans which is contrary to the interest rate charged. This can be accounted by the fact that lower the grade of the loan, higher the chances of being charged-off and hence lower will be corresponding average annual return. For fully paid loans, we observe that average return rate increases as we move to lower grade loans due to the fact that lower grade loans are charged more interest rate. For fully paid loans, a similar trend is observed within subgrade of loans. Based on this data, it is safer to invest n higher grade loans (A, B, C).

5. **What are people borrowing money for (purpose)? Examine how many loans, average amounts, etc. by purpose? Do loan amounts vary by purpose? Do defaults vary by purpose? Does loan-grade assigned by Lending Club vary by purpose?**

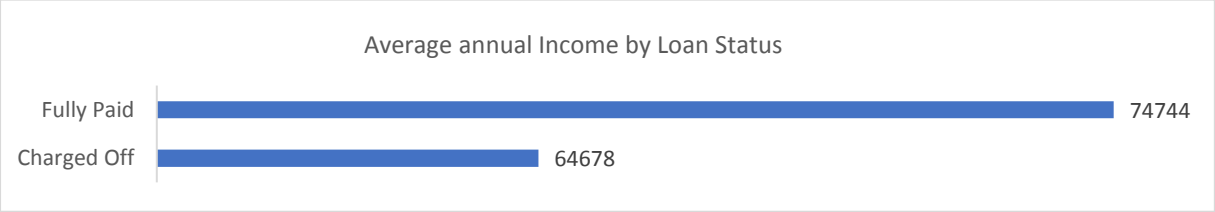
The purpose of loan is categorized into 13 different categories as 'credit card', 'small business', 'major purchase', 'vacation', 'home improvement', 'wedding' etc. The first three categories having the highest number of loans as shown below are "Debt Consolidation", "Credit Card" and "Home Improvement".



The loan amount varies by purpose having a similar distribution as the number of loans by purpose.

6. **Consider some borrower characteristics like employment-length, annual-income, fico-scores (low, high). How do these relate to loan attribute like, for example, loan_amout, loan_status, grade, purpose, actual return, etc.**

The total number of loans and proportion of fully paid loans is highest for those having 10+ years of employment. The average annual income varies with the loan grade, lower grade loans have lower average income as compared to higher grades. Grade A loans have an average income 90K as against 54K for grade G loans. In addition to above, fully-paid loans have higher average income as compared to charged-off loans. There is little correlation between loan amount and the average annual income.



7. **Generate some (at least 3) new derived attributes which you think may be useful for predicting default., and explain what these are. For these, do an analysis as in the questions above (as reasonable based on the derived variables).**

We can derive 3 new variables from the dataset which can be useful for predicting default.

- Annualized instalment to Income ratio
  This ratio gives the repayment power of the customer. Lower the ratio, higher the power to repay. Higher the income, higher the repayment power. This can be calculated by annualized return divided by the declared annual income.

- Loan amount to Total current balance ratio
  This ratio can help us evaluate the credit worthiness of the customer. Higher the total current balance, higher the credit worthiness. This can be calculated by total loan amount divided by the total balance currently in all accounts of customer. The idea is to include only those accounts where balance is non-zero.

- Delinquency score
  This score gives us to estimate the chances of a customer defaulting on loan. This is a product of two values, "whether or not the account is currently delinquent" and "no. of times customer has missed a payment in last 2 years". This score estimates the strength of customer defaulting.

2.b) Summarize your conclusions and main themes from your analyses

In the given data, 13.8% of all borrowers have defaulted and the percentage of defaulters increases as we move from Grade A to Grade G. Thus, the credit profile of borrowers decreases from Grade A to G. This is also visible in the loan amounts and interest rates. Borrowers in Grades A, B & C have more sanctions and lower interest rates whereas other grades have fewer sanctions and the average interest rate is higher.

When we compare loan terms, we observe that there is no inherent pattern but, the average Actual term of Grade A borrowers is less than 3 years. Annual Return is also better for Upper Grade borrowers owing to the credit profile.
By exploring the dataset, it can be concluded that investors that are risk averse must invest in Upper Grade borrowers that have been employed for longer periods. Whereas investors looking out for higher reward with higher risk must invest in lower grade loans.

2.c) Are there missing values? What is the proportion of missing values in different variables? Explain how you will handle missing values for different variables. You should consider what he variable is about, and what missing values may arise from – for example, a variable monthsSinceLastDeliquency may have no value for someone who has not yet had a delinquency; what is a sensible value to replace the missing values in this case? Are there some variables you will exclude from your model due to missing values?

We have 146 total variables in data. The biggest issue with the data is that large number of variables has only "NA" as their unique value. We need to remove these variables from our analysis. There are a total of 84 variables as such. Out of the 84 variables, 58 has a fill rate of more than 50% with "NA" as one of their values. We removed these variables from our analysis. The new dataset has 88 variables. Out of these 88 variables, only 6 has missing values. And the miss rate is lower than 0.065%. So, we need not remove these variables completely from our analysis. Instead, we can impute these with their mean.

Data Leakage (Q3)

Consider the potential for data leakage. You do not want to include variables in your model which may not be available when applying the model; that is, some data may not be available for new loans before they are funded. Leakage may also arise from variables in the data which may have been updated during the loan period (i.e., after the loan is funded). Identify and explain which variables you will exclude from the model.

In the given data, variables linked to the performance of a loan (i.e., behaviour of a borrower) for loans given by Lending Club will be known after the amount is funded and therefore cannot be included for assessing new borrowers. Variables like amount funded by investors & issue date will not be known for a potential borrower.

Similarly, quantities like remaining outstanding principal, interest received & any variable related to collections will be updated during the loan tenure.

The Table below lists the variables that need to be excluded –

Sr. No.	Data Leakage Variable	Description	Reason to Exclude
1	funded_amnt	The total amount committed to that loan at that point in time.	This Variable will not be available for new loans
2	funded_amnt_inv	The total amount committed by investors for that loan at that point in time.	This Variable will not be available for new loans
3	issue_d	The month which the loan was funded	This Variable will not be available for new loans
4	out_prncp	Remaining outstanding principal for total amount funded	This Variable is updated during the loan Period

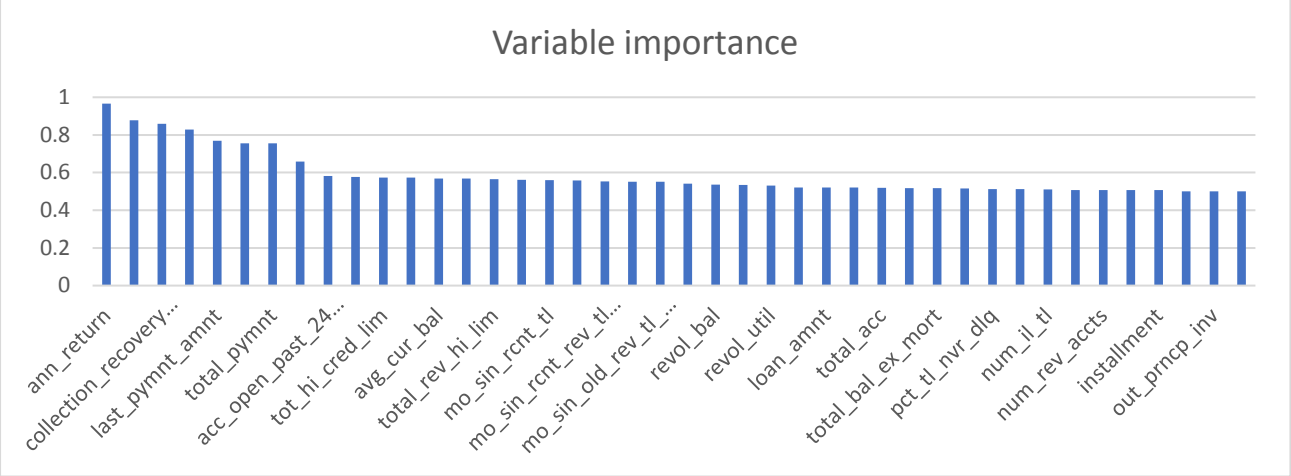
Sr. No.	Data Leakage Variable	Description	Reason to Exclude
5	out_prncp_inv	Remaining outstanding principal for portion of total amount funded by investors	This Variable is updated during the loan Period
6	total_pymnt	Payments received to date for total amount funded	This Variable is updated during the loan Period
7	total_pymnt_inv	Payments received to date for portion of total amount funded by investors	This Variable is updated during the loan Period
8	total_rec_prncp	Principal received to date	This Variable is updated during the loan Period
9	total_rec_int	Interest received to date	This Variable is updated during the loan Period
10	total_rec_late_fee	Late fees received to date	This Variable is updated during the loan Period
11	recoveries	post charge off gross recovery	This Variable is updated during the loan Period
12	collection_recovery_fee	post charge off collection fee	This Variable is updated during the loan Period
13	last_pymnt_d	Last month payment was received	This Variable is updated during the loan Period
14	last_pymnt_amnt	Last total payment amount received	This Variable is updated during the loan Period
15	next_pymnt_d	Next scheduled payment date	This Variable is updated during the loan Period
16	collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections	This Variable is updated during the loan Period
17	loan_status	Current status of the loan	This Variable will not be available for new loans
18	acc_now_delinq	The number of accounts on which the borrower is now delinquent.	This Variable is updated during the loan Period
19	tot_coll_amt	Total collection amounts ever owed	This Variable is updated during the loan Period

Most variables that will be required to assess a borrower will either be self-declared or be available in a Credit report of the potential borrower. The credit report will have information about past borrowings and will be available for a potential loan applicant.

Variable Importance (Q4)

Do a univariate analysis to determine which variables (from amongst those you decide to consider for the next stage prediction task) will be individually useful for predicting the dependent variable (loan_status). For this, you need a measure of relationship between the dependent variable and each of the potential predictor variables. Given loan-status as a binary dependent variable, which measure will you use? From your analyses using this measure, which variables do you think will be useful for predicting loan_status?

Since the dependent variable is categorical, correlation won't be useful to a univariate analysis between numerical and categorical variables. Instead, we can leverage the AUC curve to determine the important variables which have strong predictive power for our model. We build different decision tree models using a single variable each time and determine the area under the ROC curve. Out of all the variables, "Annual Return" seems to be most predictive with an AUC score of 0.97. Out of all the 88 variables that we have in our new dataset, we have 42 variables with AUC score of more than 0.5. These 42 variables are most useful for building our variables.



Part B: we will next develop predictive models for loan_status.

5. Develop decision tree models to predict default.

(a) Split the data into training and validation sets. What proportions do you consider, why?

We have split our cleaned dataset randomly into three sets: Training Set, Cross-validation set, Testing set. We have built a static cross-validation set instead of dynamic k—fold set as in the end we have zeroed down to only two models to validate on. We trained our models on the training data using different combinations of model parameters and evaluated the models on the cross-validation set. For final evaluation of the best model, we used our testing dataset. We've used the following split for the three sets:

Training set – 70% | Cross Validation set – 15% | Test Set – 15%

Training set – 69991 records | Cross Validation set – 14981 records | Test Set – 14981 records

(b) Train decision tree models (use both rpart, c50) [If something looks too good, it may be due to leakage – make sure you address this]. What parameters do you experiment with, and what performance do you obtain (on training and validation sets)? Clearly tabulate your results and briefly describe your findings. How do you evaluate performance – which measure do you consider, and why?

We've built our models using both rpart and c5.O packages. We have excluded any leakage variables while performing data cleaning. Our final cleaned dataset that has been used for building the model has 43 variables with 99874 data points.

Out of different parameters that rpart takes as arguments, we have experimented with a variety of them, such as minsplit, cp, split, loss. We have not pre-pruned out tree and are doing post pruning after growing the tree fully.

Minsplit:- We have tried different values of this parameter, mainly 30,50 and 100. Keeping all other parameters constant, we found the best value of minsplit at 30. Since we are not pre-pruning our tree and growing it to full to find the best cp for the tree size, 30 seemed the optimal option for our parameter.

CP:- We've used CP=0 so as to grow the tree fully, and then find the optimal value of CP based on xerror and xstd.

Split:- Since 'gini' is a large margin classifier, we find the best fit of our model using 'gini' as our split criteria. 'Information Gain' was also used, but we found the optimal split on 'gini'.

Loss:- We have incorporated loss matrix to penalize for the misclassification since our data has highly imbalance target variable. However, we have used loss matrix to build our second model when the first model was giving high false negative rate even after fine tuning our model.

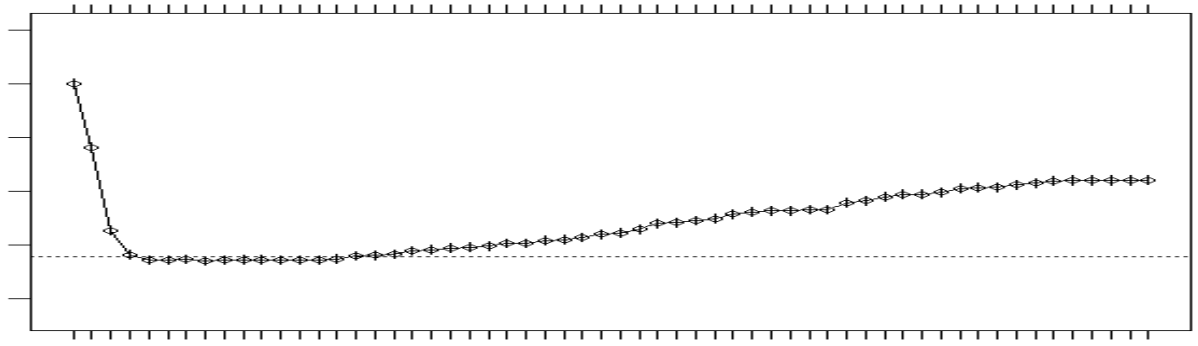
Raw Model:

We built our first model without pre-pruning, allowing the tree to grow fully and then based on the CP table, we picked the best value of CP to prune the tree. We also experimented with different combinations of parameters discussed above and found the optimal value of them. We picked minsplit = 30, split = "gini", CP=0, loss = 0.

Model 1:

```
DT1 <- rpart(loan_status ~., data=train_data, method="class", parms = list(split = "gini"), control = rpart.control(cp=0, minsplit = 30))
```

For the model above and the CP plot below, we found the optimal values of CP to be 0.0012 for our model.



We then pruned our model which has now 17 splits and then we looked at how does our model perform on validation dataset. We have identified the 'No-Default' class as our class of interest. Below are our results:-

Confusion Matrix:

Train Data:

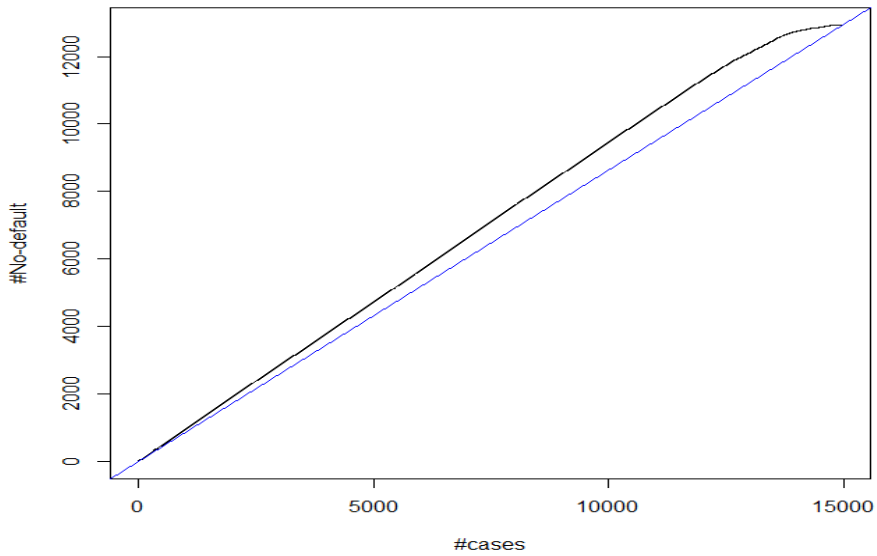
	Actual	
Predicted	**Default**	**Non-Default**
Default	4347	977
Non-Default	5214	59373

Cross-Validation Data:

	Actual	
Predicted	**Default**	**Non-Default**
Default	889	224
Non-Default	1165	12703

As per the confusion matrix above, we can deduct that the model is giving a very high false positive rate. This looks like a result of our target class being highly imbalanced. We have 13% cases of negative class and 87% of positive class. In order to fine tune better, we can over sample our negative class or use the loss matrix as one of the parameters. We also looked at model's lift chart, decile lift table and ROC curve on validation dataset.

Lift Curve:

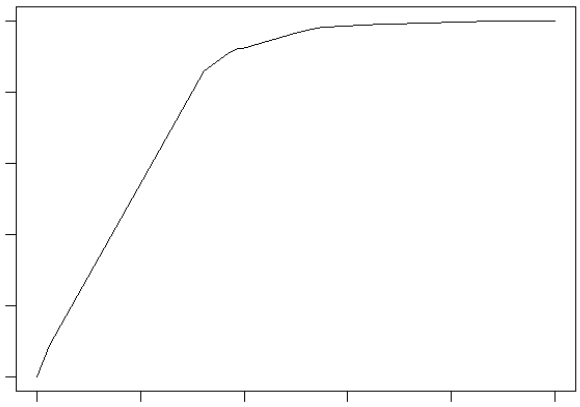


Decile Lift Table:

bucket	count	noDefaults	nodefRate	cumnoDefRate	lift
1	1499	1431	0.954636	0.954636	1.106321
2	1498	1408	0.93992	0.93992	1.089266
3	1498	1409	0.940587	0.940587	1.090039
4	1498	1416	0.94526	0.94526	1.095455
5	1498	1429	0.953939	0.953939	1.105512
6	1498	1409	0.940587	0.940587	1.090039
7	1498	1407	0.939252	0.939252	1.088492
8	1498	1384	0.923899	0.923899	1.070699
9	1498	1168	0.779706	0.779706	0.903596
10	1498	466	0.311081	0.311081	0.36051

As observed from the lift chart and decile lift table, we observe that our model is performing slightly better as compared to the no-model scenario. We went ahead to look at the ROC curve of this model with our class of interest being 'No-Default'.

ROC Curve:



Our ROC curve also points to a slightly better situation as compared to no-model scenario. As observed in the ROC curve and mentioned above, since, we have a high false positive rate, our model is performing sub-optimally.

We next build our 2nd model with an additional parameter of loss matrix and then we looked at its CP table to find the optimal value of CP in order to prune the tree.

Model 2:

```
DT2 <- rpart(loan_status ~., data=train_data, method="class", parms = list(split =
"gini",loss=matrix(c(0,10,30,0))), control = rpart.control(cp=0, minsplit = 30))
```

We have included a misclassification cost of 10 and 30 in order to better our false positive rate. For the model above and the CP plot below, we found the optimal values of CP to be 0.0019 for our model. We then pruned our model which has now 13 splits and then we looked at how does our model perform on validation dataset. We have 'No-Default' class as our class of interest. Below are our results:-

Confusion Matrix:

Train Data:

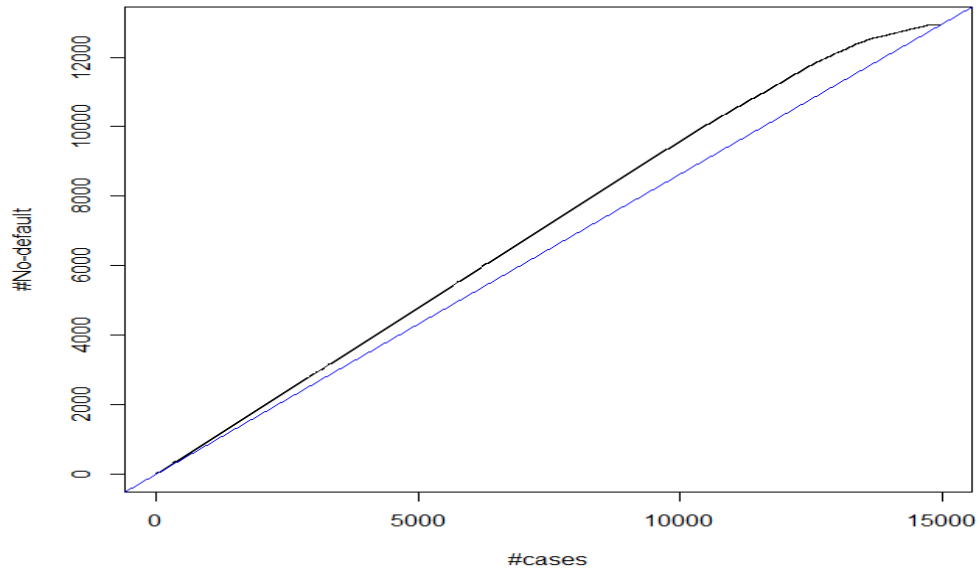
	Actual	
Predicted	**Default**	**Non-Default**
Default	6158	4932
Non-Default	3403	55418

Cross-Validation Data:

	Actual	
Predicted	**Default**	**Non-Default**
Default	1278	1020
Non-Default	776	11907

As observed from the evaluation matrices above, the performance of the model on the negative classes has increased. Specifically, the model's precision has increased from 52% to 65%. We also looked at our 2nd model's lift chart, decile lift table and ROC curve on validation dataset to establish more concretely the 2nd model's performance on validation set.

Lift Curve:

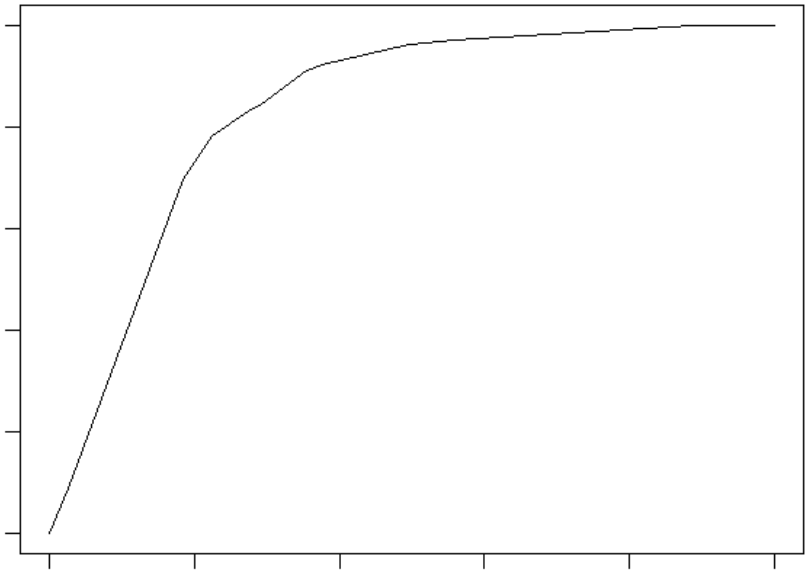


Decile Lift Chart:

bucket	count	noDefaults	nodefRate	cumnoDefRate	lift
1	1499	1431	0.954636	0.954636	1.106321
2	1498	1408	0.93992	0.93992	1.089266
3	1498	1409	0.940587	0.940587	1.090039
4	1498	1416	0.94526	0.94526	1.095455
5	1498	1429	0.953939	0.953939	1.105512
6	1498	1409	0.940587	0.940587	1.090039
7	1498	1407	0.939252	0.939252	1.088492
8	1498	1384	0.923899	0.923899	1.070699
9	1498	1168	0.779706	0.779706	0.903596
10	1498	466	0.311081	0.311081	0.36051

As observed from the lift chart and decile lift table, we observe that our model is performing slightly better as compared to the no-model scenario. However, this does not prove the fact that Model 2 is performing better that Model 1 and hence we went ahead to look at the ROC curve of this model with our class of interest being 'No-Default'.

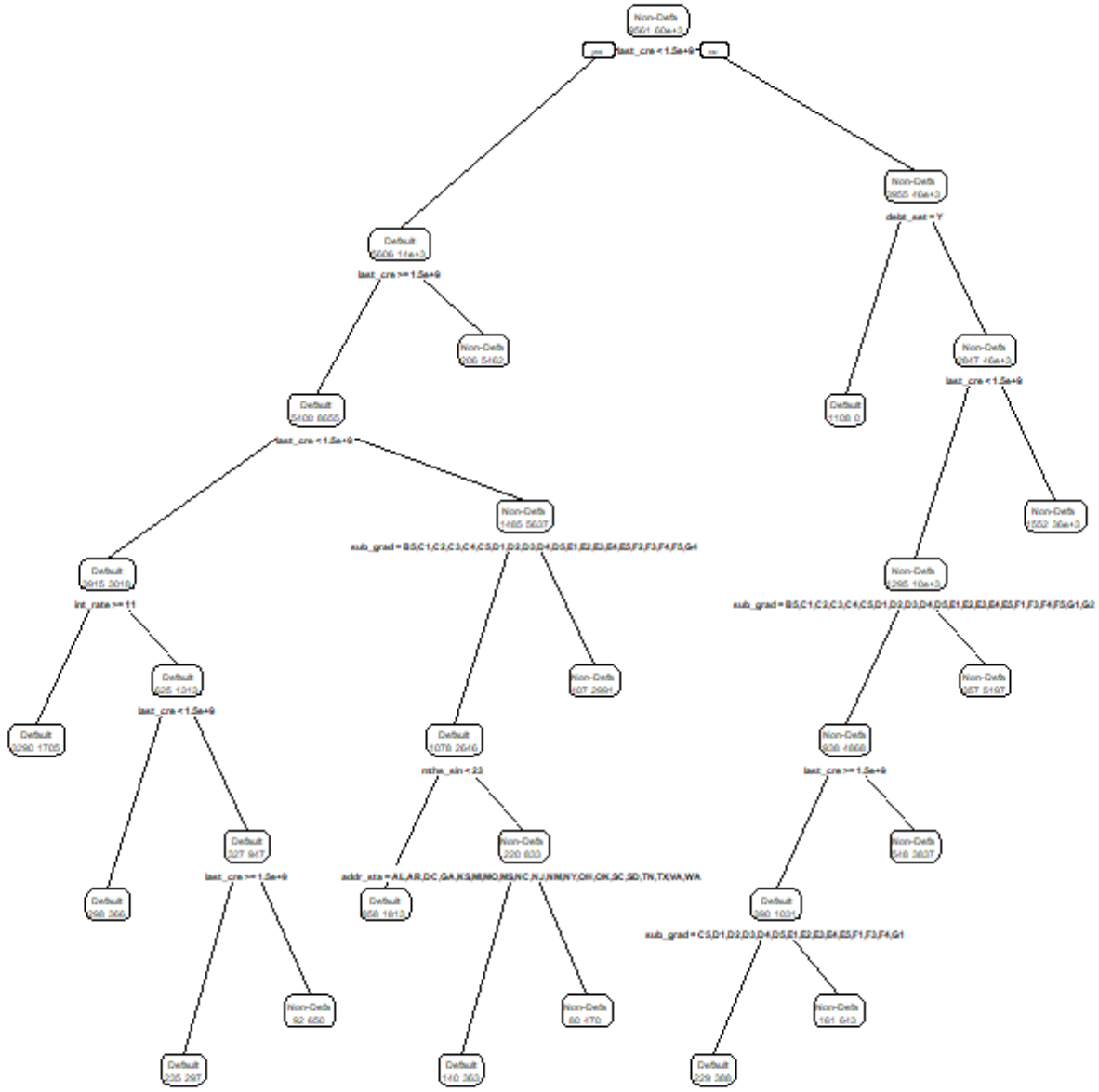
ROC Curve:



Takeaway: *A quick comparison of our model 1's ROC curve with Model 2's ROC Curve validates the fact that Model 2's performance is evidently better that Model 1. This is also visible with the increase the model's precision. Hence, our decision tree Model 2 is the optimal model for classification purpose with the class of interest being 'No-Default/Fully-Paid'.*

(c) Identify the best tree model. Why do you consider it best? Describe this model – in terms of complexity (size). Examine variable importance. How does this relate to your uni-variate analyses in Question 4 above? Briefly *describe* how variable importance is obtained (the process used in decision trees).

The best tree model that we built after using different combinations of parameter testing, model fine tuning and evaluation on validation dataset is our 2nd model i.e. Model 2. We considered it based on its precision, F1 Score and ROC curve. From all different combinations of model, this model's performance on the final test set has the best precision score, F1 score and ROC curve. Below is a glimpse of our final tree model.



Our chose complexity parameter for the is 0.0019 with minimum number of cases for a split at a node is 30. The tree is optimal is size which doesn't overfit the model as observed from the cross-validation error.

The variable importance for our model in decreasing importance is as below :-

last_credit_pull_d, debt_settlement_flag, int_rate, sub_grade, grade, bc_open_to_buy, total_bc_limit, total_rev_hi_lim , initial_list_status.

As compared with our initial univariate analysis, 'last_credit_pull_d', 'int_rate', 'grade' are same as we found out from our initial variable importance analysis.

Variable importance in decision tree is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the variable.

6. Develop a random forest model. (Note the 'ranger' library can give faster computations). What parameters do you experiment with and does this affect performance? Describe the best model in terms of number of trees, performance, variable importance. Compare the performance of random forest and best decision tree model from Q 5 above. Do you find the importance of variables to be different?

Which model would you prefer, and why? For evaluation of models, you should include confusion matrix related measures, as well as ROC analyses and lifts. Explain which performance measures you focus on, and why.

For random forest, we have split our cleaned dataset randomly into two sets: Training set and Testing set. Random forest utilizes bootstrap sampling and provides good approximation of test error on out-of-bag (OOB) samples. Hence, we do not need to reserve any chunk of data for validation testing and in turn utilize all that data for training the model. We trained our models on the training data using different combinations of model parameters and evaluated the models using OOB error and the test set. For random forest models, we've used the following splits:

Training set – 80% | Test Set – 20%

Training set – 79899 records | Test Set – 19975 records

We've built our models using both 'randomForest' and 'ranger' packages. We have excluded any leakage variables while performing data cleaning. Our final cleaned dataset that has been used for building the model has 43 variables with 99874 data points.

<u>Models using 'randomForest':</u>

We built our first model using randomForest package and without providing any parameters. By default, randomForest built a model with N (number of tress) = 500 and mtry (number of variables to possibly split at in each node) = square root of the number variables (rounded down) = 6

Model 1:

```
rf.randomForest <- randomForest(loan_status ~., data=train_data)
```

The model provided an aggregate OOB error at each value of 'N'. We then looked for a value of 'N' for which OOB error was minimum, which came out to be N=369. Also, the corresponding value of 'mtry' came out as mtry=6.

We then built second model using randomForest package, now tuning it by providing N=369, mtry= 6 and importance = TRUE as additional parameters.

Model 2:

rf.randomForest.Tuned <- randomForest(loan_status ~., data = train_data, ntree = 369, mtry = 6, importance = TRUE)

The model provided an aggregate OOB error at each value of 'N'. We observed that the value of 'N' for which OOB error was minimum now came out to be N=172

To further analyze the effect of N over OOB error, we plotted N vs OOB error for both models as follows,

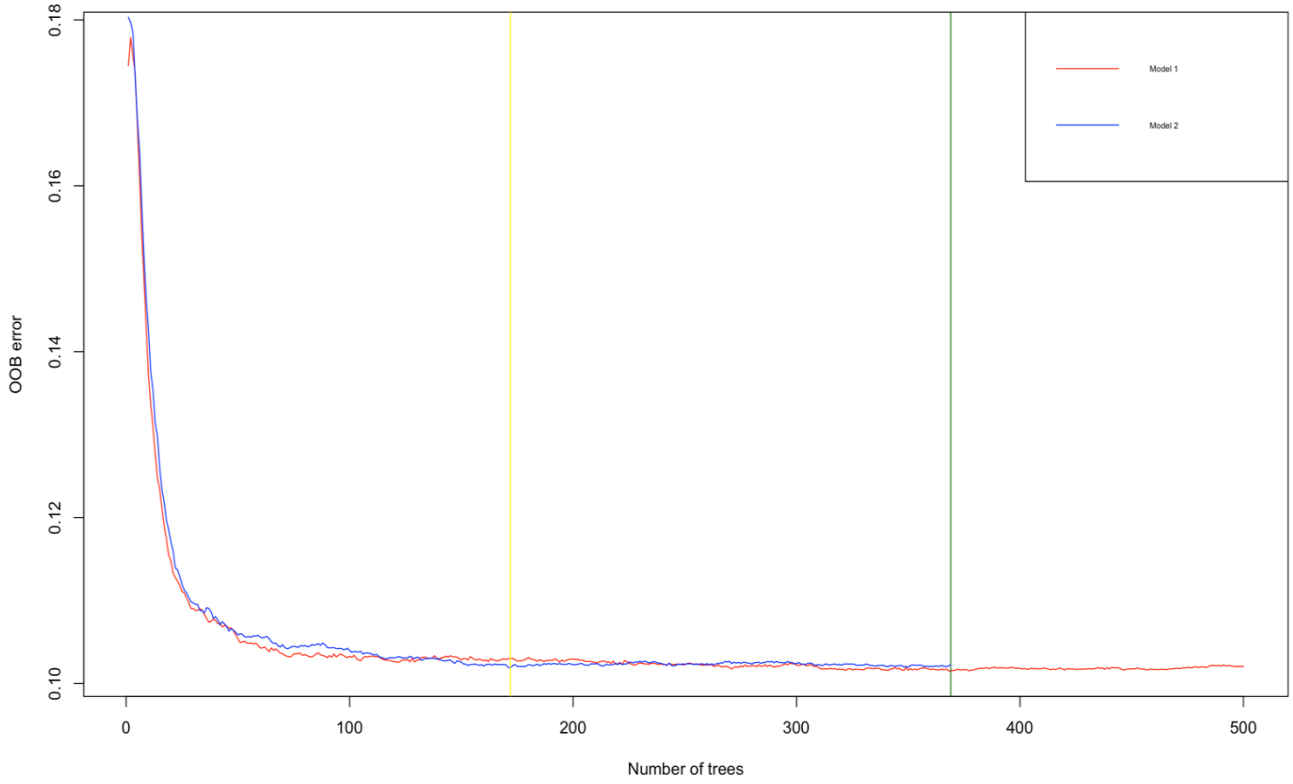


Figure 1. N vs OOB error

We looked at how our models perform on train and test datasets. We have identified 'Non-Default' as our class of interest. Our evaluation results are as follows,

Model 1

Confusion Matrix:

Train Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	3548	7395
Non-Default	757	68199

Accuracy = 0.8979

Test Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	947	1823
Non-Default	171	17034

Accuracy = 0.9001

Model 2

Confusion Matrix:

Train Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	3494	7449
Non-Default	720	68236

Accuracy = 0.8977

Test Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	933	1837
Non-Default	156	17049

Accuracy = 0.9002

Takeaway:

As per the confusion matrix above, we can deduct that the models are giving overall good accuracy. From Figure 1, we can see that for lower values of N, OOB error is quite high but then it steeply decreases as N > 100. OOB error remains fairly consistent with increase in N after 100.

Models using 'ranger':

We built our third model using ranger package and without providing any parameters. By default, ranger built a model with N (number of tress) = 500 and mtry (number of variables to possibly split at in each node) = square root of the number variables (rounded down) = 6

Model 3:

```
rf.ranger <- ranger(loan_status ~., data=train_data)
```

The model provided an aggregate prediction error of 0.102 with N = 500, mtry = 6.

We then built our fourth model using ranger package, now tuning it by keeping N = 500, mtry= 6 and but importance = 'permutation' as additional parameter.

Model 4:

```
rf.ranger.Tuned <- ranger(loan_status ~., data = train_data, num.trees = 500, mtry = 6, importance =
'permutation')
```

The model also provided an aggregate prediction error of 0.102 keeping N = 500, mtry = 6 but adding importance = 'permutation' parameter.

We looked at how our models perform on train and test datasets. We have identified 'Non-Default' as our class of interest. Our evaluation results are as follows,

Model 3

Confusion Matrix:

Train Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	3548	7395
Non-Default	734	68222

Accuracy = 0.8982

Test Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	957	1813
Non-Default	156	17049

Accuracy = 0.9014

Model 4

Confusion Matrix:

Train Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	3563	7380
Non-Default	739	68217

Accuracy = 0.8983

Test Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	968	1802
Non-Default	169	17036

Accuracy = 0.9013

Takeaway:

As per the confusion matrix above, we can deduct that the models are performing slightly better with overall good accuracy. We tried to change variable importance parameter by using 'permutation', but observed that the performance of model compared to one with 'none' as importance parameter was almost same.

Full-grid search across range of hyper-parameters

To understand the effect a particular hyper-parameter and their combinations as whole on model performance, we did a full-grid search across range of values for following important hyper-parameters:

mtry -> values (3, 5, 7, 9)

Reasoning: The default value of 'mtry' for classification is sqrt (number of features), which in our case comes down to 6, and hence we investigated in range +- 3 from 6.

num.tress -> values (100, 150, 200, 250, 300, 350, 400, 450, 500)

Reasoning: The default value for 'num.trees' is set to 500, and from Figure 1., we can very well estimate that N>100 does not change OOB error much, hence we investigated in range 100 to 500. Anything with N>500 for our case would not sufficiently decline OOB error and may potentially lead to overfitting.

importance - > values ('impurity', 'permutation')

Reasoning: The default value is set to 'none', and we wanted to test the effect on model by setting variable importance to two most popular values used for classification.

*The total number of hyper-parameter combinations are **72**. We trained a random forest model using ranger by iterating through each value combination in grid and stored the aggregate OOB-error for that model.*

```
params_grid <- expand.grid(
            mtry      = seq(3, 9, by = 2),
            num.trees  = seq(100, 500, by = 50),
            importance = c('impurity', 'permutation'),
            OOB.error  = 0
)
for(i in 1:nrow(params_grid)) {
            rfModel <- ranger(
              formula    = loan_status ~.,
              data       = train_data,
              mtry       = params_grid$mtry[i],
              num.trees  = params_grid$num.trees[i],
              importance = params_grid$importance[i]
            )
           params_grid$OOB.error[i] <- rfModel$prediction.error
}
```

Then, the results were sorted by OOB-error (from least to most), and tabulated as follows,

mtry	num.trees	importance	OOB.error
9	450	permutation	0.095808458
9	400	permutation	0.095933616
9	400	impurity	0.096096322

mtry	num.trees	importance	OOB.error
9	350	permutation	0.096208964
9	500	permutation	0.096208964
9	350	impurity	0.09622148
9	500	impurity	0.096271543
9	300	permutation	0.096271543
9	250	impurity	0.09630909
9	250	permutation	0.096359153
9	300	impurity	0.096421732
9	450	impurity	0.096509343
9	200	impurity	0.096784691
9	150	impurity	0.09723526
9	200	permutation	0.097385449
9	150	permutation	0.097485576
9	100	impurity	0.098274071
9	100	permutation	0.098611998
7	500	impurity	0.098987472
7	400	impurity	0.099062566
7	500	permutation	0.099062566
7	350	permutation	0.099137661
7	450	impurity	0.099162693
7	250	permutation	0.099375462
7	300	impurity	0.099413009
7	250	impurity	0.099638293
7	350	impurity	0.099713388
7	450	permutation	0.099763451
7	200	permutation	0.099963704
7	200	impurity	0.100013767
7	150	impurity	0.100063831
7	300	permutation	0.100226536
7	400	permutation	0.100389241
7	150	permutation	0.100702136
7	100	permutation	0.101578243
7	100	impurity	0.102554475
5	350	permutation	0.104494424
5	350	impurity	0.104544487
5	400	impurity	0.104594551
5	450	impurity	0.104644614
5	300	impurity	0.104794803
5	400	permutation	0.104807319
5	250	impurity	0.104919961
5	500	permutation	0.104932477
5	250	permutation	0.105020088
5	450	permutation	0.105245372
5	200	impurity	0.105358014
5	150	permutation	0.105383046
5	500	impurity	0.10552072
5	200	permutation	0.105683425
5	150	impurity	0.105695941
5	100	permutation	0.106284184

mtry	num.trees	importance	OOB.error
5	300	permutation	0.106471921
5	100	impurity	0.107485701
3	100	permutation	0.11294259
3	350	impurity	0.113268001
3	200	permutation	0.113480769
3	250	permutation	0.113881275
3	300	permutation	0.114031465
3	400	impurity	0.11410656
3	500	permutation	0.11410656
3	100	impurity	0.114169139
3	350	permutation	0.114244233
3	250	impurity	0.114269265
3	500	impurity	0.114281781
3	450	permutation	0.114294297
3	450	impurity	0.114319328
3	150	permutation	0.114394423
3	300	impurity	0.114557128
3	200	impurity	0.114607192
3	150	impurity	0.114719834
3	400	permutation	0.11473235

From above table, we can infer the following about the effect of hyper-parameter on performance of model:

mtry:

A small value (less features considered when splitting at each node) will reduce the variance of the ensemble, at the cost of higher individual tree (and probably aggregate) bias. Increasing the maximum number of random features considered in a split tends to decrease the bias of the model, as there is a better chance that good features will be included, however this can come at the cost of increased variance. Also, there is a decrease in training speed when we include more features to test at each node.

num.trees:

Keeping all other hyper-parameters fixed, increasing the number of trees generally reduces model error but at the cost of a higher training time.

importance:

Keeping all other hyper-parameters fixed, model performs slightly better with 'permutation' instead of 'impurity'. This is because permutation method directly measures variable importance by observing the effect on model accuracy of randomly shuffling each predictor variable. On the other hand, the impurity importance of a feature is computed by measuring how effective the feature is at reducing uncertainty (classifiers) or variance (regressors) when creating decision trees within RFs. In summary, 'permutation' is reliable but computationally slower, while 'impurity' is fast but not reliable since it tends to be biased in certain situations (by inflating the importance of continuous or high-cardinality categorical variables).

Now, considering the minimum OOB error as our primary metric, our best model from hyper-parameter grid is one with parameters - mtry=9, num.trees=450, importance=permutation

Optimum Model

```
rf.Optimal.Model <- ranger(
        loan_status ~.,
        data = train_data,
        num.trees = rf.Model.Eval.Grid[1,"num.trees"],
        mtry= rf.Model.Eval.Grid[1,"mtry"],
        importance = rf.Model.Eval.Grid[1,"importance"])
```

The model provided an aggregate prediction error of 0.09703. We looked at how our optimum model performs on train and test datasets. We have identified 'Non-Default' as our class of interest. Our evaluation results are as follows,

Confusion Matrix:

Train Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	4329	6676
Non-Default	1077	67817

Accuracy = 0.9029

Test Data:

	Predicted	
Actual	**Default**	**Non-Default**
Default	1095 (TN)	1613 (FP)
Non-Default	253 (FN)	17014 (TP)

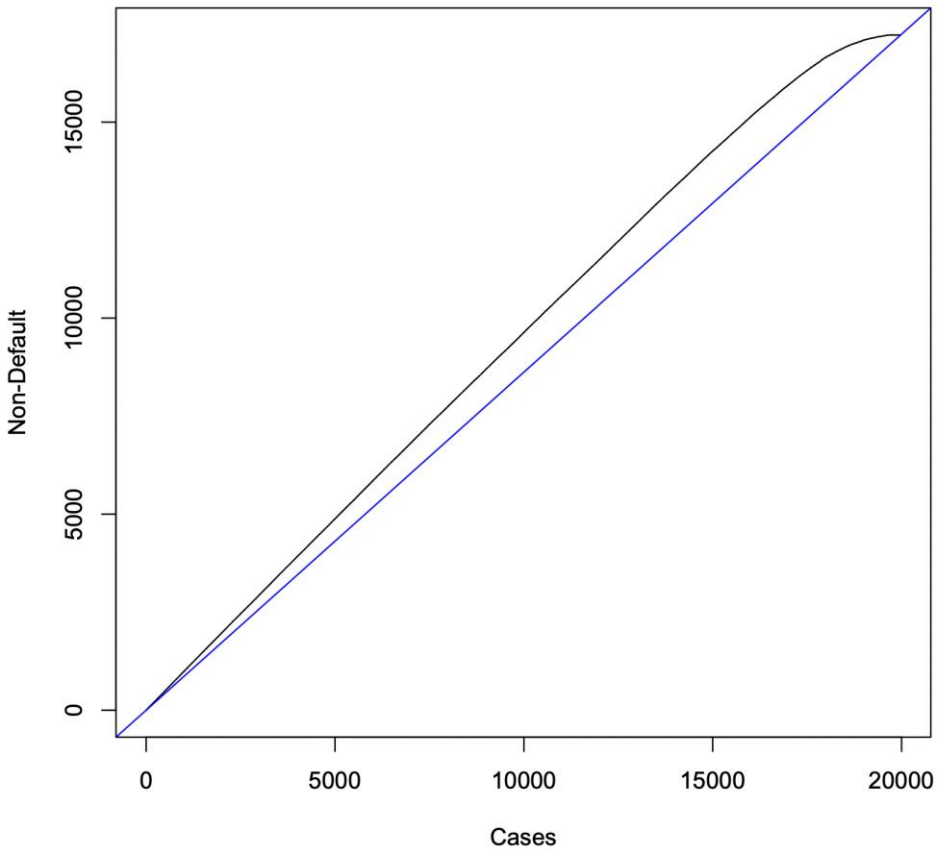
Accuracy = 0.9065

As expected, the model performs quite well with decent jump on overall accuracy. Following are the performance metrics calculated based on confusion matrix for test data:

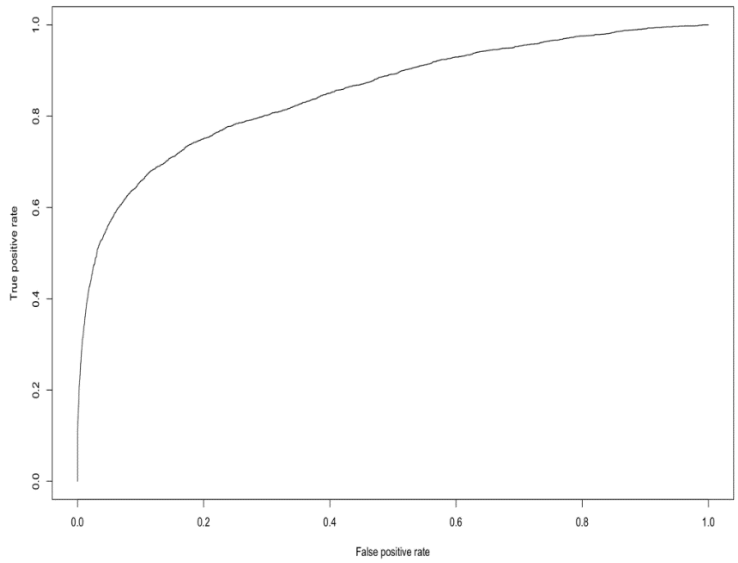
Non-Default = 1, Default = 0

True Positive Rate $_{(test)}$ = 0.9853, Precision $_{(test)}$ = 0.9134, F-Score$_{(test)}$ = 0.9479

Lift Curve:



ROC Curve:



Lift curve and ROC curve points to a slightly better situation as compared to no-model scenario.

Comparing our optimum RF model to best DT model in Q5, we can see that performance of RF model is comparable to DT model, but little better. RF model has a slightly better ROC and Lift curve and a higher F-score.

Random Forest is suitable for situations like these where we have a large dataset and interpretability is not a major concern. As random forest leverages the power of multiple decision trees and not rely on the feature importance given by a single decision tree, they can generalize over the data in a better way and provide more accurate results than a decision tree.

Our business objective here is to develop models to identify good/bad loans ('fully paid' or 'charged off'), and these models should be evaluated on basis on how well they can classify positive class "Non-Defaults". Hence, our metric on focus should be Sensitivity as it tells how apt the model is to classify given example as positive class.

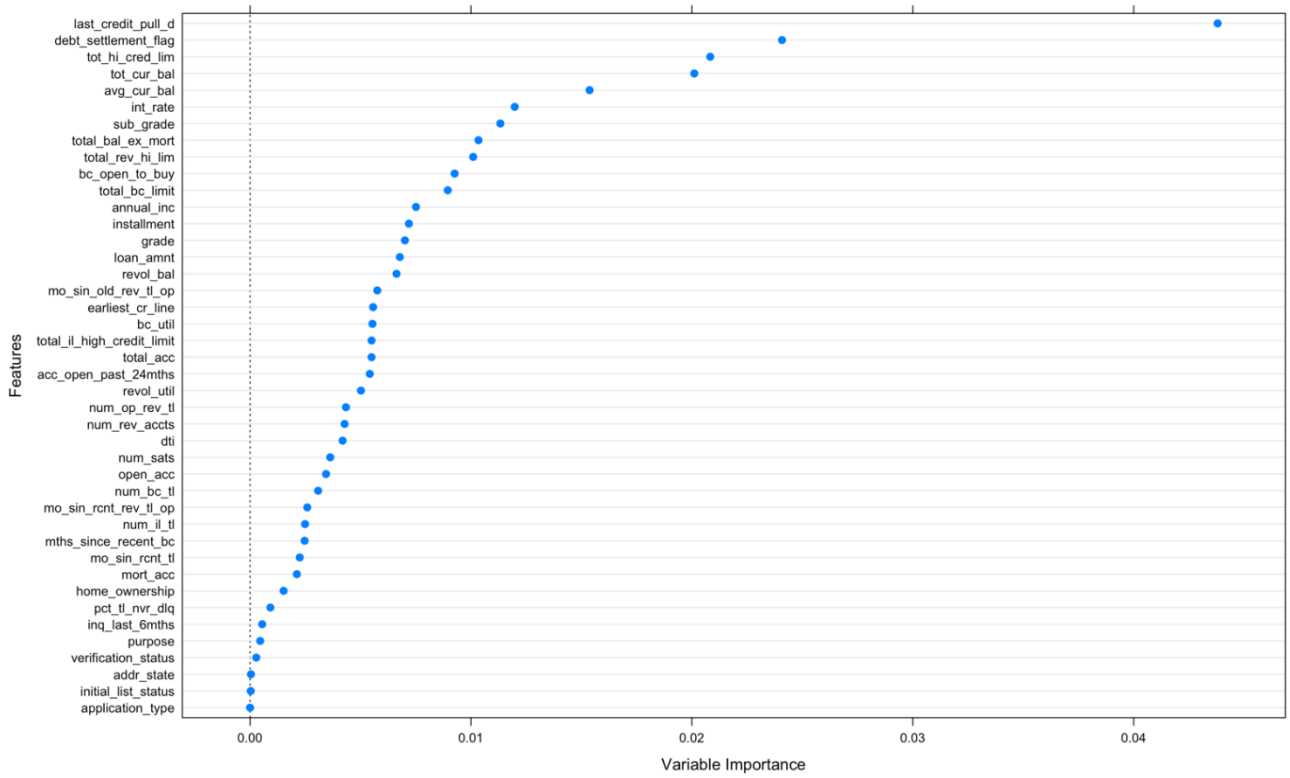
Now,

Sensitivity $_{DT}$ ***= 0.9210***

Sensitivity $_{RF}$ ***= 0.9853***

Thus, due to better Accuracy, Sensitivity & F-score, robustness of algorithm, and better generalization over unseen data, we would choose Random Forest over Decision Tree as our preferred model in this scenario.

Variable Importance



The variable importance for our model in decreasing importance is as below:

last_credit_pull_d, debt_settlement_flag, tot_hi_cred_lim, tot_cur_bal, avg_cur_bal, int_rate, sub_grade, total_bal_ex_mort, total_rev_hi_lim, and so on.

As compared with variable importance for DT in Q5, the first two prominent variables viz. 'last_credit_pull_d' and 'debt_settlement_flag' are same in order but with different importance (weights). Other variables such as int_rate, sub_grade, total_rev_hi_lim are also present but with different importance (weights).

7. The purpose of the model is to help make investment decisions on loans. How will you evaluate the models on this business objective? Consider a simplified scenario - for example, that you have $100 to invest in each loan, based on the model's prediction. So, you will invest in all loans that are predicted to be 'Fully Paid'. Key questions here are: how much, on average, can you expect to earn after 3 years from a loan that is paid off, and what is your potential loss from a loan that has to be charged off?

(i) One can consider the average interest rate on loans for expected profit – is this a good estimate of your profit from a loan? For example, suppose the average int_rate in the data is 11.2%; so after 3 years, the $100 will be worth (100 + 3*11.2) = 133.6, i.e a profit of $33.6. Now, is 11.2% a reasonable value to expect – what is the return you calculate from the data? Explain what value of profit you use.

Estimating the profit value of a loan by using the average interest rate is a reasonably good measure since it is the approximate total return on investment. The profit value can be accurately predicted by taking into account the reinvestments and inflation rates.

For the models created in Q5 & Q6, we need to use historical data to ascertain the returns. Therefore the average rate over the total data needs to be considered.

The average interest rate (grouped for non-default loans) in the data is 11.7% (whereas for Defaulting borrowers, the average interest rate is 13.9%). Now, for a three-year investment, (100 + (3

11.7)) = $135.1 i.e. a **profit value of $35.1*

(ii) For a loan that is charged off, will the loss be the entire invested amount of $100? The data shows that such loans have do show some partial returned amount. Looking at the returned amount for charged off loans, what proportion of invested amount can you expect to recover? Is this overly optimistic? Explain which value of loss you use.

*As observed in Part A of the assignment, Defaulted loans have a negative annual return. This happens because in most cases the principal amount itself is unpaid. To calculate the loss value we have considered the return over 3 years (i.e. amount paid -amount funded). The average annual loss is -35.9% i.e for every $100 invested in a default loan, **$35.9 will be lost** (or investor will be left with $64.1 on a $100 investment). In case of Loss value, we have not considered the average interest rate as we assume that certain borrowers try to make payments even after they default. The average interest may not capture this possibility.*

(iii) You should also consider the alternate option of investing in, say in bank CDs (certificate of deposit); let's assume that this provides an interest rate of 2%. Then, if you invest $100, you will receive $106 after 3 years (not considering reinvestments, etc), for a profit of $6.

For any loan predicted as a "Default", the profit will be $6, since the amount will be invested in a CD. (For all models)

(a) Compare the performance of your models from Questions 5, 6 above based on this. Note that the confusion matrix depends on the classification threshold/cutoff you use. Evaluate different thresholds and analyze performance. Which model do you think will be best, and why?

Note – Calculations are done based on Probability of Default

Model 1 DT – Profit: $415130.8 (For a default threshold of 0.5)

Model 2 DT – Profit: $408272.3 (For a default threshold of 0.3)

Model RF Optimal – Profit: $ 411734.6

For Model 1, When the threshold is reduced to 0.3, multiple Non-default loans are predicted as defaults and therefor, the profit decreases. Similarly, when the threshold is increased to 0.7 the profit decreases since defaulting loans that were predicted as are now predicted as defaults are now predicted as Fully paid borrowers. This causes the Loss value to be multiplied with a higher number and this subsequently brings the profit down.

Threshold values of 0.4 & 0.6 do not influence the profit drastically. Therefore, the highest profit is seen at threshold of 0.5 and seems to be the best model

For Model 2, At a threshold of 0.29, the profit generated is $408272.3 at 0.3 the value increases to $410771.6 the value decreases if increased above 0.4.

For a ranger model, we cannot manipulate the thresholds directly. So, at the default threshold the profit is Lower than DT1 but higher than DT2

(b) Another approach is to directly consider how the model will be used – you can order the loans in descending order of prob(fully-paid). Then, you can consider starting with the loans which are most likely to be fully-paid and go down this list till the point where overall profits begin to decline (as discussed in class). Conduct an analyses to determine what threshold/cutoff value of prob(fully-paid) you will use and what is the total profit from different models. Also compare the total profits from using a model to that from investing in the safe CDs. Explain your analyses and calculations. Which model do you find to be best and why. And how does this compare with what you found to be best in part (a) above.

Note – Calculations are done based on Probability of Non-Default

For Model 1 DT, when we observe the table with the probability of fully paid loans arranged in descending order, we find that after 13754 rows (Total test data is 14982 rows), the profits begin to decline due to erroneous predictions. The Probability of having a non-defaulting borrower is 0.61. So, if we change our threshold to 0.65, then the profit slightly increases (by approximately $400)

Model 1 DT – Profit: $415130.8 (Default Threshold of 0.5)

Model 1 DT – Profit: $415512.4 (Threshold of 0.65)

In Model 2 DT, we find that the profits decline after reaching a certain point, we observe that the model has classified several Non defaulting borrowers as defaulters and therefore, the capital that could be invested in these loans is invested in CDs. If we lower the threshold slightly, we can include many such borrowers. Thus, increasing profits by almost $3000.

Model 2 DT – Profit: $408272.3 (Default Threshold of 0.71)

Model 2 DT – Profit: $412876 (Threshold of 0.65)

In the random Forest model, Profits are better at lower thresholds. The model correctly identifies most non-defaulting borrowers.

Model RF Optimal – Profit: $ 348841 (At threshold of 0.5)

Model RF Optimal – Profit: $ 379288.9 (At threshold of 0.1)

The 2 factors driving the decision will be the costs in the confusion matrix & the threshold. In some models, the ***Cost of Omission*** *(Opportunity Cost of not securing a fully paying borrower) is less whereas in certain models, the* ***Cost of Commission*** *(Cost lost servicing a defaulting borrower) is less. It's the combination of both these costs that defines the final profit value. The threshold value can cause changes in the profit as changing the threshold may change the distribution of cases in a confusion matrix.*

On comparing the models, the second approach combined with the second DT model seems like the best (Most profitable) option for the given data set. Although, it depends on an investor based on their risk appetite.