**Lending Club Predictive Model**

**Introduction:**

The data provided in the CSV file is the past data where the users were considered creditworthy or not. There was lot s of criteria specified for each user. Some of the main criteria’s are Loan Purpose, Loan term, Home Ownership, Loan Amount, FICO Credit score, and annual income in others. Good prediction should be able to clear distinguish between creditworthy and Uncreditworthy. But not all the cases are same and the data patterns are such that it is quite impossible to do so. It happens either the full data set based on which the decision were made are missing or there are times when with almost similar criteria some people are specified creditworthy and Uncreditworthy. Thus it is quite challenging to find the proper patterns, which exists in the data.

**Data Pre-processing:**

This is most import step in any Machine-learning algorithm and most of the time is spent here. The data was quite good and did not have too much data missing. However, for some 52 rows there were missing values. I ran the R code in the file **Pass\_1.rmd** its output is **Pass\_1.docx** and it does following.

1. Find the missing data.
2. Find any features, which could be a candidate for factor columns.
3. Find any factor columns, which have say, more than a given number of levels. This is useful because after some levels some models don’t work. Also, the graphs wouldn’t look nice. Also, the data might not be enough to represent all the levels. Thus some of the levels will have very few values.
4. Find the correlation between columns.
5. It also finds out the unusual data e.g. there might be some factor columns where one level is way too much underrepresented. We could combine these with other levels base don independence among levels wr.t. Class or remove these as these could cause issues especially when we will split the data in test and train data sets.
6. It does a test for finding which columns are related to each other or independent of each other. This includes correlation for numeric columns and a chi square independence test for categorical variables. For relation between numeric and categorical variables I convert the numeric variable to categorical using chi merge and then chi-square of independence method is applied. This process doesn’t rely on the class column and entirely between features. Thus, it could be used on whole data set.
7. Then it generates the graphs of individual columns as well as all combination of graphs. It could generate loads of graphs thus I always sample 20k rows and use these 20 k rows for graphs as it makes easy to graph the data on smaller set and at same time provides me enough information. I usually look for the trend especially where there is quite large distinct between given classes.

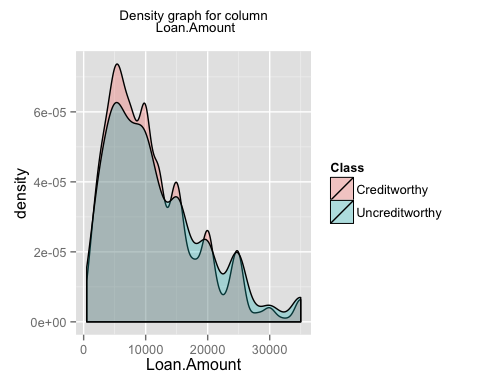
Once I had run the script. I had enough information about the data. However, looking at the graphs it was kind of impossible to distinguish the classes as the density graphs of two classes for numerical columns were overlapping each other like almost covering each other. However, the graphs between 2 columns were showing some trend but still it was not something that could have given the idea that there are clear pattern between classes.

Also, there were some missing values and there were some factor columns for which the representations from few levels were far too few and these could have created problems especially when you split the data. It also showed me that some of the continuous columns could be converted to ordered factor columns.

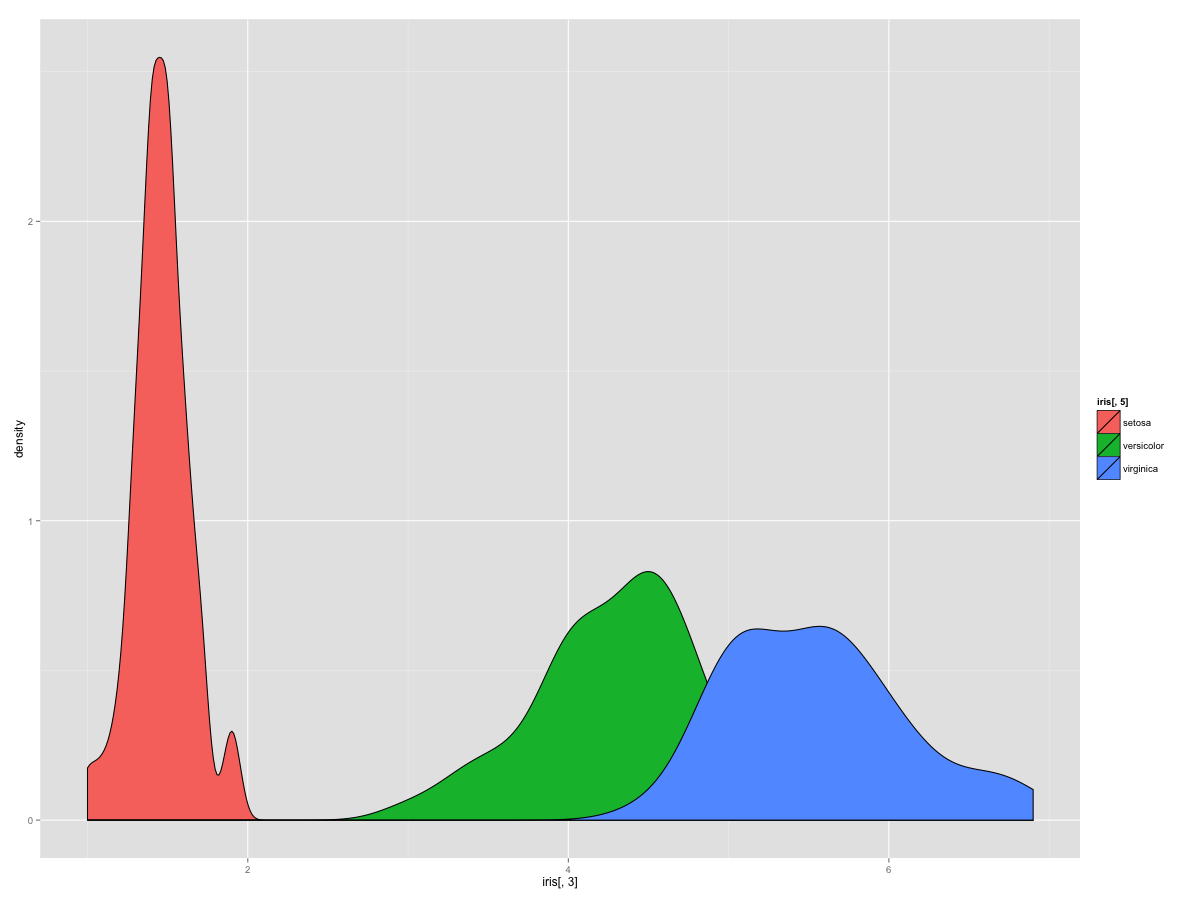
Below are some interesting graphs, which would be useful for data analysis and feature selection.

Density Graph for Loan.Amount:

It doesn’t distinguish much between 2 classes. This was confirmed using the single column logistic regression model as well as with Random forest feature selection.

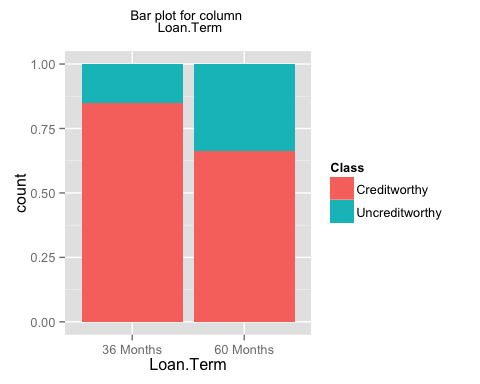


Compare this with the graph based on Iris data and for Petal.Length column. Here one class is quite distinctive and remaining 2 classes have slight overlap and in fact this column alone is enough to get quite high accuracy, recall and precision. I was looking for these sort of trends but these are quite difficult in he real world data especially in the credit scoring data as some of the data might not be captured.



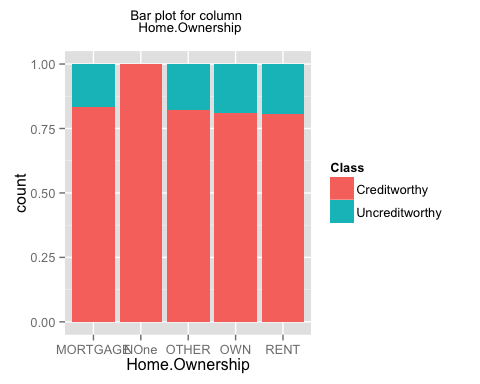
Bar Chart for Loan Term:

This seems somewhat discriminative feature and RF and LR feature importance confirm it as well.



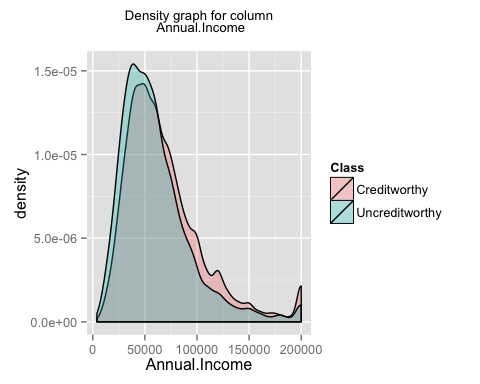
Bar Chart for Home.Ownership:

This seems somewhat discriminative feature especially we could say that for None value we have creditworthy and among others it is quite difficult to distinguish. However, issue here is that we just have 4 data points for None and thus it shouldn’t be discriminative. I have included it here to show that sometimes graphs can conceive you. RF and LR feature importance confirm it as well. I could have used the graph, which has the height of the bar as the count, but in that case there are many cases where one level has way too much data and thus you cannot see any of the patterns properly.



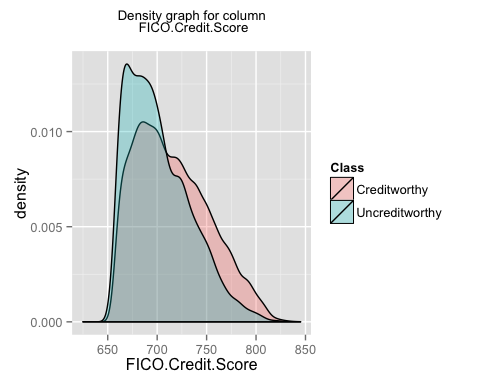
Density Graph for Annual Income.

This seems somewhat discriminative feature and RF and LR feature importance confirm it as well.



Density Graph for FICO Credit Scoring:

This seems somewhat discriminative feature and RF and LR feature importance confirm it as well.



**Feature Extraction:**

My next task was to find useful features, which I could use in my models. In this case there are not too many features so large dimensionality was not an issue but I wanted to remove the columns which were not discriminative enough.

Thus, I removed the missing data and then I split the data in 3 sets. One was test set, which had the 30% of the data. Then from remaining data I kept around 7% of data for validation and remaining 63% for the training.

Then, I merged the levels based on independence w.r.t. Class column. Thus, it is a supervised problem and **doing so on full dataset might have caused the bias.** Thus, I checked the independence only on training data and then based on independence I merged the columns in raining, validation data and train data. Train data and validation data was not used in checking independence.

Then, I ran graphs again as I had different levels in some of the factor columns. Finally, I ran the feature selection script. **Script ran only on training data.** Script name is **lv\_pass\_2\_feature\_selection\_graphs.Rmd**.

Output is the word document.

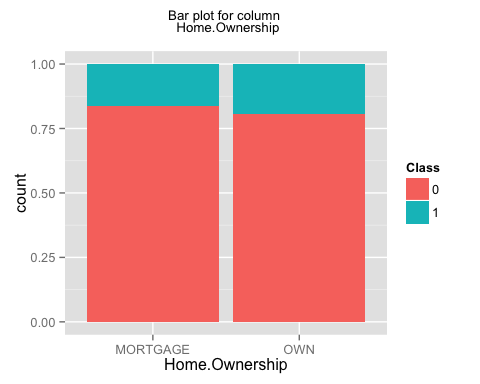
**lv\_pass\_2\_feature\_selection\_graphs.docx**

The process is as follows.

1. It first tries to merge the levels of a column, which are independent. This step shouldn’t be used everywhere. I am using it because the some of the factor columns have very few values and it will cause issues when we will split the data. In some cases a level might be there in test data but not in train data. Maybe with k fold we wont have this issue but for that too there must be at least 2 rows and still there is no guarantee.
2. It checks the independence of all columns w.r.t. Class column. Continuous columns are converted to factor columns using a combination of unsupervised (equal frequency discretization) and supervised (Chi Merge discretization) process. This gave which columns are independent of the class. But this doesn’t mean that we can throw the columns away as they might be useful with interaction with another columns.
3. Then, it ran the random forest model and ranked the top 20 features based on accuracy measure.
4. Then, It ran the logistic regression by taking one column at a time.
5. Them, It ran the logistic regression but I used 2 columns and their interaction. I could have done the simple interaction as well. This was to find which column interaction should I use in my final LR.
6. Then, It ran a decision tree on each column by using cp value of 0.001. This was to find which columns are discriminative.
7. Them, it ran the decision tree but with 2 columns at a time. Again with same cp value.

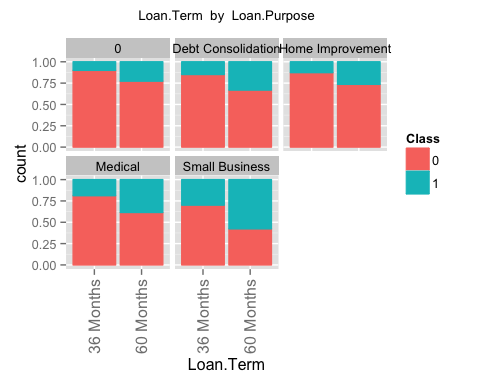
Some of above methods **are very computationally expensive** and shouldn’t be run on large data or when we have large number of features. But Lending Club dataset was not so big and thus I have run these methods.

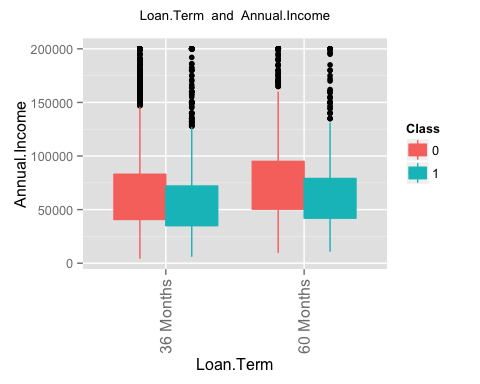
Below are some graphs after merging of levels. This is for Home Ownership, which was discussed earlier. Now you can see that the feature is not that important however OWN has slightly more uncreditworthiness ratio. Here we cannot interpret it as 2 categories OWN and Mortgage as we do not know which category has what extra category in them. Here we lose somewhat the interpretation power,



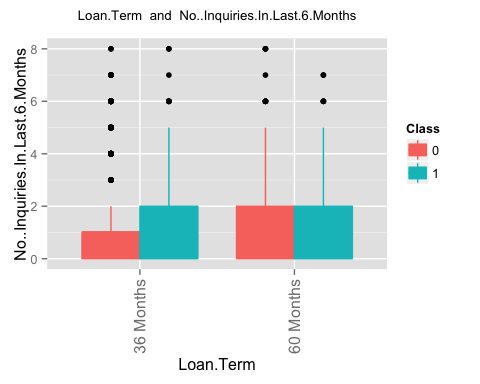
Here are some important interaction graphs, which are based on top interaction terms, which gave highest AUC when used alone in LR model. There were other graphs as well which I found using graphs rather than using the AUC but those had the same issue as the homeownership bar chart explained earlier where graphs shows a pattern but number of observation are quite low.

These are self-explanatory.











**Model:**

**Performance Measure Metrics:**

This is another very important factor in modelling and you have to choose it right. Usually, people prefer the accuracy in most of the cases but there are cases where recall and AUC is being used as well.

In cases where the classes are not so much balanced AUC is a much better predictor of performance rather than accuracy.

In our case if we were to use accuracy as measure then classifying everything as creditworthy would have given us the accuracy around 83%-84% as we have around 83-84% of creditworthy data. Also, recall is good measure in these cases but just alone recall is not good enough. If you wanted to have the recall of uncreditworthy class to be the best then we could have classified everything as uncreditworthy. But it would not have been ideal case.

Here, I will be using AUC as well as recall along with GINI and KS Scores. Usually people use GINI coefficient [8] and KS Scores in the credit scoring.

AUC, and roc curve gives us an idea where to have our cut-off. I am using the cut-off value for confusion matrix where the KS was realized [6] i.e. where the difference between TPR and FPR was maximised. I could have used other cut-off values based on what is my ultimate goal. I could have calibrated the cutoff values on the validation set and then could have used it on the test set.

**Class Imbalance problem:** In this case we have around 16% from positive class (i.e. Uncreditworthy) and thus we have some sort of bias as most of the models tries to decrease the misclassification. Although it is not that we have a rare class case here but this imbalance means some bias in models towards predicting most of data as negative class. Thus, I will use smote as well in one of my model. I will not try different combinations for smote settings.

Column No Of Credit lines and Total No of credit lines are correlated and correlation coefficient is around 0.70. I could remove one of them but l will go ahead with both of them.

You will see that I have used validation set as well as the test set. This should be used especially when you want to calibrate the model parameters.

In my case I did not use the calibration at all. Thus, I ran the model on test set as well but when you do calibrate the parameters you should not touch the test set unless and until you have finalized your models and then only you should run it on test case. If you will use the test set during calibration you will add bias into your models and it might not generalize well.

Finally here are our models.

**Model 1:**

Simple model using plain Logistic Regression.

**Script:**

**Part3\_modelling\_first\_lr\_latest. Rmd**.

**Formula:**

‘Class ~. ‘

**Results:**

On Validation set:

AUC

## [1] 0.713752

GC

## [1] 0.427504

KS

## [1] 0.3459363

cutoffvalue

## [1] 0.1670585

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 875 86  
## 1 542 227

recall

## [1] 0.7252396

acc

## [1] 0.6369942

ROC Curve



On Test set:

AUC

## [1] 0.7149375

GC

## [1] 0.429875

KS

## [1] 0.3312571

cutoffvalue

## [1] 0.18973

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 4189 483  
## 1 1883 861

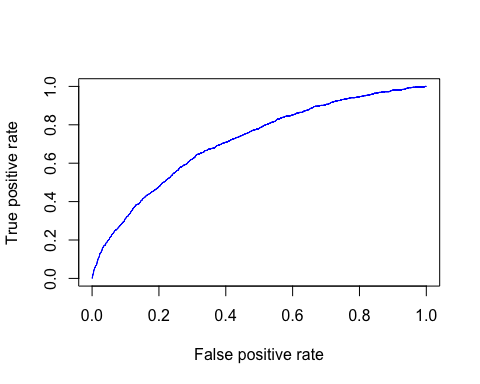
recall

## [1] 0.640625

acc

## [1] 0.6809601

ROC Curve



**Observation:**

This looks a good model and with the data and no good discriminative feature the auc value looks fine to me and the recall and accuracy are almost as good as any other model out there.

**Model 2:**

Here, I tried to convert all the continuous columns, which has very few distinct values. Then I merged the levels based on the chi square independence test. Then again I used the same formula.

**Script:** **Part3\_modelling\_first\_lr\_latest\_smote.Rmd**.

The data has the class imbalance problem and thus I wanted to use the smote method to oversample the rare class and under sample the common class. After smote the new data looked like below.

table(train\_smt$Y)

##   
## 0 1   
## 9884 8472

**Formula:**

‘Class ~.

**Results:**

AUC

## [1] 0.6916989

GC

## [1] 0.3833979

KS

## [1] 0.3119198

cutoffvalue

## [1] 0.256966

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 777 75  
## 1 640 238

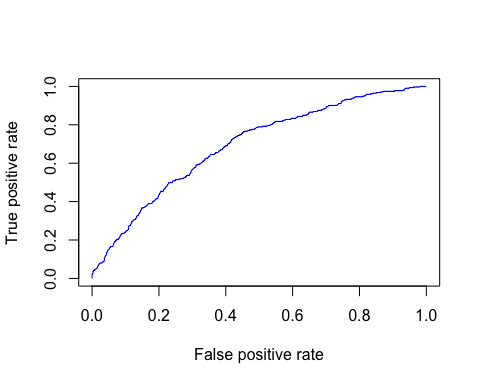
recall

## [1] 0.7603834

acc

## [1] 0.5867052

ROC Curve



AUC

## [1] 0.7022514

GC

## [1] 0.4045029

KS

## [1] 0.3093562

cutoffvalue

## [1] 0.2756175

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 3672 398  
## 1 2400 946

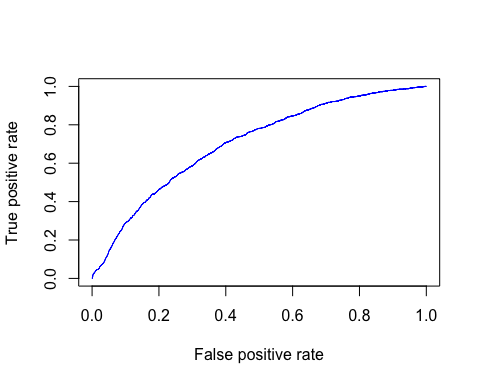
recall

## [1] 0.703869

acc

## [1] 0.6227077

ROC Curve



**Observation:**

There was no improvement over the actual model. The validation set results are worse but on test data it improved a lot.

**Model 3:** Here, I tried to run the random forest. I removed column “No..Of.Credit.Lines” from the model as it has a correlation of around 0.67 with column “Total No..Of.Credit.Lines”.

**Script: Part3\_modelling\_top\_k\_colBased\_on\_RF.Rmd**

**Formula:**

'Class ~ .

On Validation Set:

AUC

## [1] 0.7213728

GC

## [1] 0.4427457

KS

## [1] 0.3418823

cutoffvalue

## [1] 0.198

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 917 98  
## 1 500 215

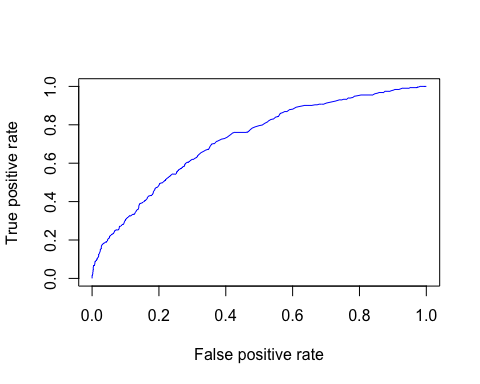
recall

## [1] 0.686901

acc

## [1] 0.6543353

ROC Curve



On Test set:

AUC

## [1] 0.7163036

GC

## [1] 0.4326072

KS

## [1] 0.3175701

cutoffvalue

## [1] 0.184

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 3661 387  
## 1 2411 957

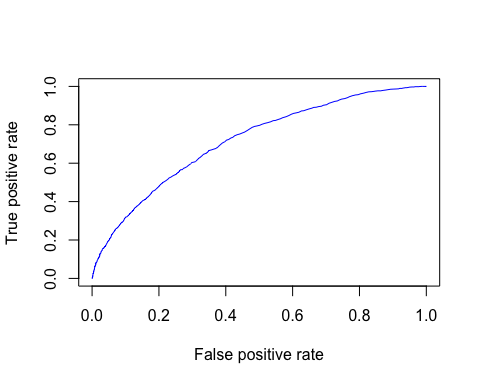
recall

## [1] 0.7120536

acc

## [1] 0.6227077

ROC Curve



**Observation:**

This is the best model in terms of the auc both on the validation set as well as on the test set. However, the test set showed a lot of difference in the recall and accuracy whereas on validation set it was quite good.

**Model 4:**

Here, I wanted to see whether I could add some of the interaction terms based on the logistic regression I ran for interaction of columns. I just took top 10 of those values based on the auc score for each term when that term was only term in the model.

**Script: Part3\_modelling\_add\_interaction\_col\_try\_lr.Rmd**

**Formula:**

frml1 <- 'Class ~ . + Loan.Purpose:FICO.Credit.Score + Loan.Purpose:Use.Of.Credit.Line + Loan.Term:FICO.Credit.Score + Loan.Term:Annual.Income + Loan.Term:Use.Of.Credit.Line + Annual.Income:FICO.Credit.Score + Loan.Term:Loan.Purpose + FICO.Credit.Score:No..Inquiries.In.Last.6.Months + No..Inquiries.In.Last.6.Months:Use.Of.Credit.Line + FICO.Credit.Score:Total.Number.Of.Credit.Lines + Loan.Term:No..Inquiries.In.Last.6.Months'

**Result:**

On Validation Set:

AUC

## [1] 0.7133642

GC

## [1] 0.4267284

KS

## [1] 0.3458596

cutoffvalue

## [1] 0.1718012

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 893 90  
## 1 524 223

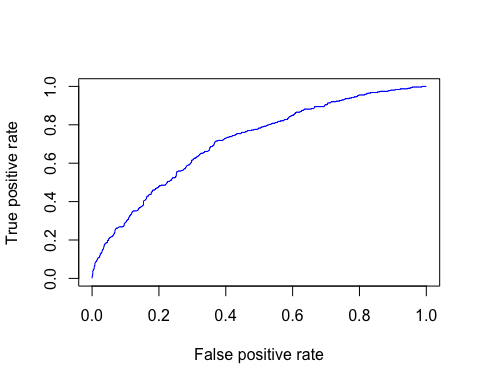
recall

## [1] 0.7124601

acc

## [1] 0.6450867

ROC Curve



On Test Set:

AUC

## [1] 0.7152062

GC

## [1] 0.4304124

KS

## [1] 0.3307718

cutoffvalue

## [1] 0.1818469

Confusion Matrix (rows predicted, columns actual)

## 0 1  
## 0 4046 452  
## 1 2026 892

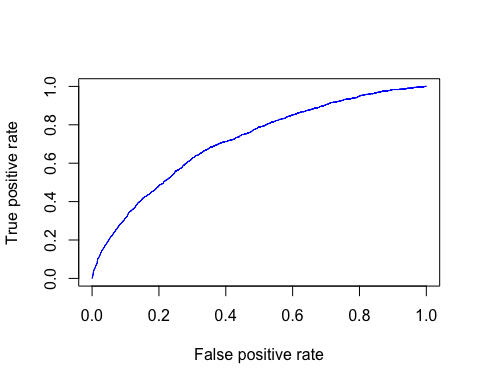
recall

## [1] 0.6636905

acc

## [1] 0.6658576

ROC Curve



**Observation:** This model is very similar to the original model. Thus, you can choose of either models but before that run it using say k folds and see how will it perform. Also, try to remove the extra columns in the model, which are not use full and see if it improves over the existing model.

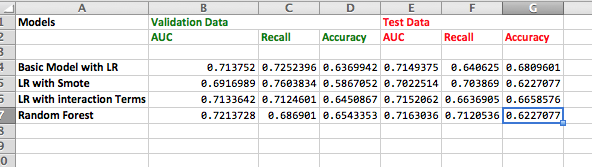
**Conclusion:**

The data set has doesn’t have enough discriminative or predictive features. Usually, running a random forest gives you an idea of how good your features are. In this case even random forest performance was not good. However, these are as per industry standard where you see recall and accuracy around 0.65.

Also, I feel that there are some columns, which are missing especially the information about individual e.g. the age, sex, working status and marital status etc. Also, the debt to income ratio column is not quite clear to me as what it indicates and it could have been better if we had actual debt details and then maybe we could have derived the new debt to income ratio based on old debt and new loan amount. Also, it would have been much better to talk to people and see why some people were rejected a loan.

Also, I could have used a KNN and see if a person is rejected for a loan, what was the outcome for say nearest 5 neighbours. I have used similar things in my code but I have not properly analysed it and see what else could make good features or is there some sort of pattern there.

Model performance summary is below.



I have not specified that I would prefer one method to other. Also, I have not done statistical significance testing say for AUC between different methods. However, I would do a statistical significance testing for AUC and then would select the best method. I would need to do this either using k fold or using say k runs on validation set and then I would select best method and then run it on test set to see how it performs.

**Further work:**

However, I could have done following to improve the AUC or recall or accuracy further or would do in future based on time I will have.

1. I did not use a cost matrix where we could penalize the model more if it makes mistakes on one class as compared to others. I could have tried this as well.
2. Use the logistic regression interaction columns, which are, not say in top list when used individually.
3. I could have removed the individual columns which were not discriminative enough either based on feature selection process or after I had created my first Model.
4. As there are very few columns, I could have run brute force methods to find the all individual as well as interaction columns for a pair.
5. I could have tried other models like GBM, Neural Net (logistic regression) or Normal Neural Net along with the random forest or SVM models.
6. I could have tried different SMOTE settings
7. I could have used a different sort of method to tackle the class imbalance problem. E.g. I could have divided the common calls data in train data into say equal number of folds which has similar size as the rare class and then I could have run the model for each fold and the rare class instances and then finally averaged out the probabilities for these models to create the final model. I have initiated this method but did not do averaging out part. See the script. Part3\_modelling\_add\_interaction\_col\_try\_lr\_multfolds.Rmd
8. I could have used extra columns say based on summary statistics of the interaction columns e.g. for Loan.Term and Fico.Credit.Score and I could have used the value in the columns as probability of being uncreditworthiness. Similarly, I could have added for another interaction columns.
9. I could have added 3 new columns where I could have added say the knn based prediction with k=1,3,5 and then use these in final models.
10. I could have played with different setting of parameters for LR as well as RF.

**Code:**

All my code is in github directory. It is attached as a zip file as well in submission. It used some of the functions, which I have created, and these are generic functions. I have provided the script names, which I had used, in respective parts in bold above. It will need a list of libraries before it could be executed. The code is written in generic terms and thus to see how the code works you might need to dig a bit deep. Basic steps would be

1. Unzip the file LV.zip.
2. Open R Studio. Then open a project and browse to LV/LV.Rproj.
3. Now build the LV.Rproj in R studio.
4. Then run pass\_1.Rmd. It mght take some time.
5. Run “lv\_pass\_2\_feature\_selection\_graphs.Rmd”. However, note that it will create a different train, test and validation set than what I have used here as it will overwrite those files. Although, results should be quite similar. It will generate various graphs and will show which features are significant. Details are given in Feature Selection part. It might also take some time to run.
6. Now you can run the models as mentioned in modeling section. The file names are given in the modelling section for each model.
7. Finally, if you face any issues. Let me know and I can demonstrate it.

I have loaded the train, test and valid data csv files to github as well so that these can be reproduced. But if you run “lv\_pass\_2\_feature\_selection\_graphs.Rmd” then the files will be overwritten and the results can not be reproduces as it is but would be quite similar.

**The code path is**

<https://github.com/darshanmeel/LV/tree/master/R>

**Required libraries are**

Rpart

randomForest

ROCR

LV ( This is my project where I have defined multiple functions to make my coding simple)

discretization

ggplot2

unbalanced

Caret

AppliedPredictiveModelling

**References:**

1. Dataset was provided to me as part of the interview process for Logical Glue. They have prepared it base don the data from Lending Club.

<https://www.lendingclub.com/>

Data Set is below:

[https://s3-eu-west-1.amazonaws.com/logicalgluebucket/interview/LendingClub.csv](https://s3-eu-west-1.amazonaws.com/logicalgluebucket/interview/LendingClub.csv" \t "_blank)

The data set cannot be used for commercial purposes without getting permission from Logical Glue team.

2. Guide to Credit Scoring: <http://cran.r-project.org/doc/contrib/Sharma-CreditScoring.pdf>

3. Applied predictive Modelling <http://appliedpredictivemodeling.com/>

4. Data Mining with R <http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR/>

5. Kaggle competitions <http://www.kaggle.com/>.

6. <http://www.statsoft.com/textbook/credit-scoring#Classic_credit_scoring>

7. Whole web & Google and Stack overflow.

8. <http://www.business-school.ed.ac.uk/waf/crc_archive/2013/18.pdf>