**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 users. 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** 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.

**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**.

The process is as follows.

1. 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.
2. Then, it ran the random forest model and ranked the top 20 features based on accuracy measure.
3. Then, It ran the logistic regression by taking one column at a time.
4. 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.
5. Then, It ran a decision tree on each column by using cp value of 0.001. This was to find which columns are discriminative.
6. 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.

Now I had the graphs and all of this data to see which columns were important.

**Model:**

**Performance Measure Metrics:**

This is another very important method 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.

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 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.

**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. I had tried the smote but that did not improve the performance much. I have not included that model here.

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 as correlated features wont cause any issue in the logistic regression.

Finally here are our models.

**Model 1:**

Simple model using plain Logistic Regression.

**Script:**

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

**Formula:**

‘Class ~.

**Results:**

AUC

[1] 0.7138692

> GC

[1] 0.4277385

> KS

[1] 0.3304646

Confusion Matrix ( rows predicted, columns actual)

0 1

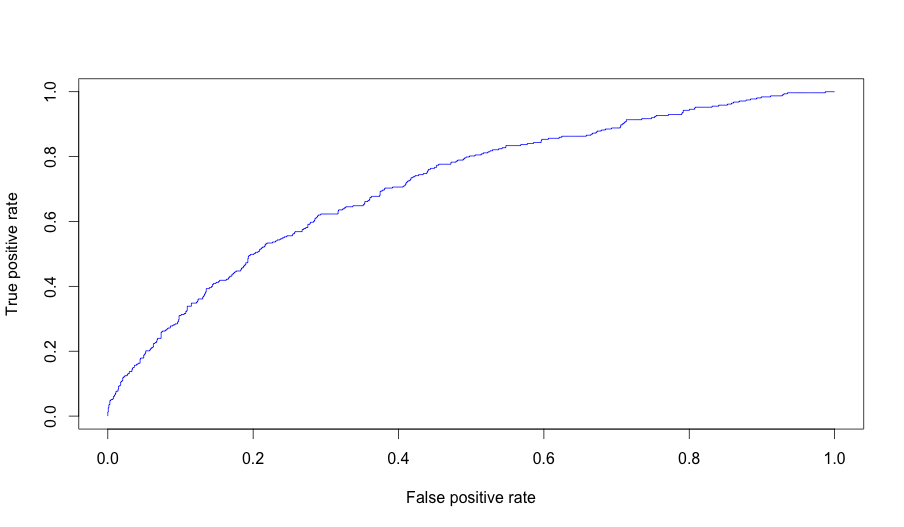
0 1007 120

1. 410 193

> recall

[1] 0.6166134

ROC Curve:



**Observation:**

This looks a good model and with the data and no good discriminative feature the auc value looks fine to me.

**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\_convert\_num\_to\_fact\_try\_lr.Rmd**.

**Formula:**

‘Class ~.

**Results:**

> AUC

[1] 0.7062532

> GC

[1] 0.4125064

> KS

[1] 0.3205139

Cutoff:

[1] 0.1655261

Confusion Matrix (rows predicted, columns actual)

0 1

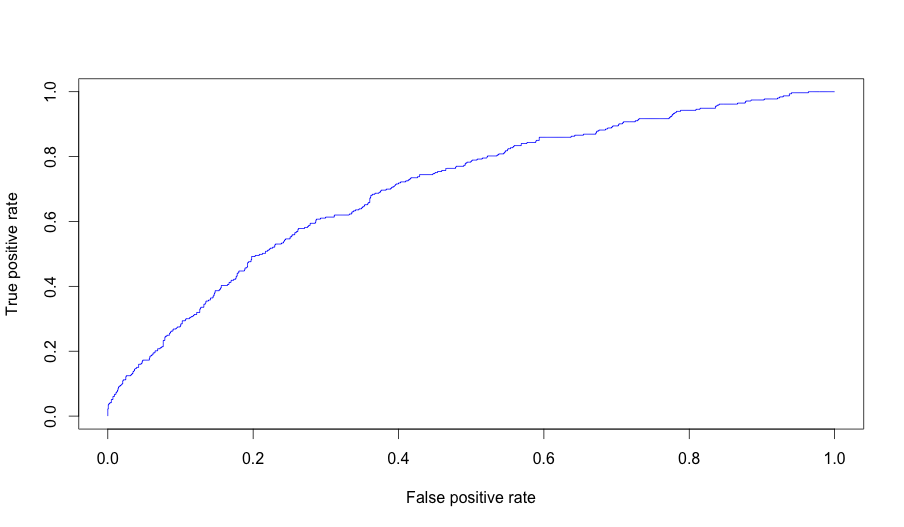
0 883 96

1 532 217

> recall

[1] 0.6932907

ROC Curve:



**Observation:**

There was no improvement over the actual model.

**Model 3:** Here, I wanted to see whether I could add some of the interaction terms based on the decision trees I got from feature extraction. I wanted to use the decision tree generated for a pair of columns. Then I used these terms in my formula.

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

**Formula:**

'Class ~ . + Loan.Purpose:Loan.Term + Loan.Purpose:Loan.Amount + Loan.Purpose:No..Delinquencies.In.Last.2.Years + Loan.Purpose:Use.Of.Credit.Line'

**Results:**

> AUC

[1] 0.7171746

> GC

[1] 0.4343492

> KS

[1] 0.3388137

> cutoffvalue

[1] 0.1475286

Confusion Matrix (rows predicted, columns actual)

0 1

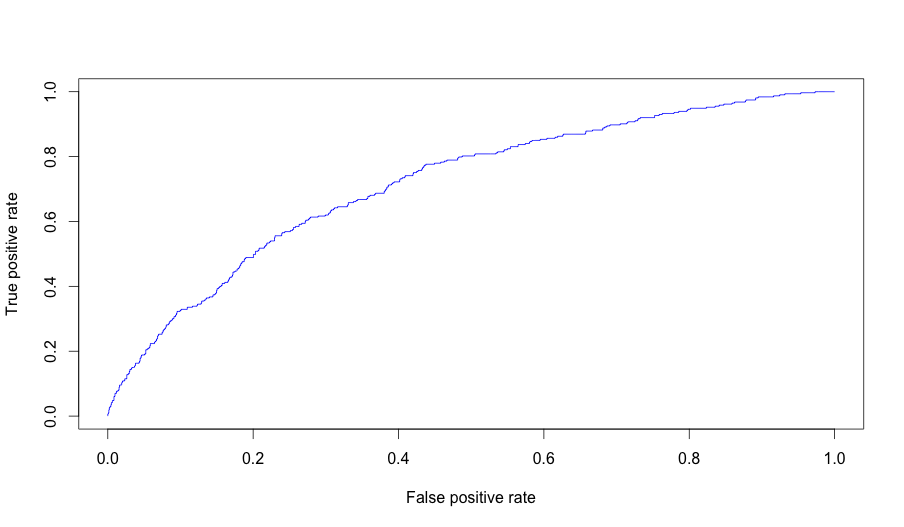
0 797 71

1 620 242

> recall

[1] 0.7731629

ROC Curve:



**Observation:**

Performance improved over actual model. But I did not do any statistical significance test whether improvement was a significant one. I could have done it as part of k fold validation.

**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 6 of those values based on the p values. I could have used the coefficient ordering as well along with the p values.

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

**Formula:**

frml1 <- 'Class ~ . + No..Of.Credit.Lines:Use.Of.Credit.Line + No..Adverse.Public.Records:Loan.Application.Description

+ No..Of.Credit.Lines:No..Of.Public.Record.Bankruptcies + Loan.Term:Loan.Application.Description + Home.Ownership:Loan.Application.Description + Earliest.Credit.Line.Opened:Use.Of.Credit.Line'

**Result:**

> AUC

[1] 0.7183425

> GC

[1] 0.4366851

> KS

[1] 0.3326968

> cutoffvalue

[1] 0.1839355

Confusion Matrix (rows predicted, columns actual)

0 1

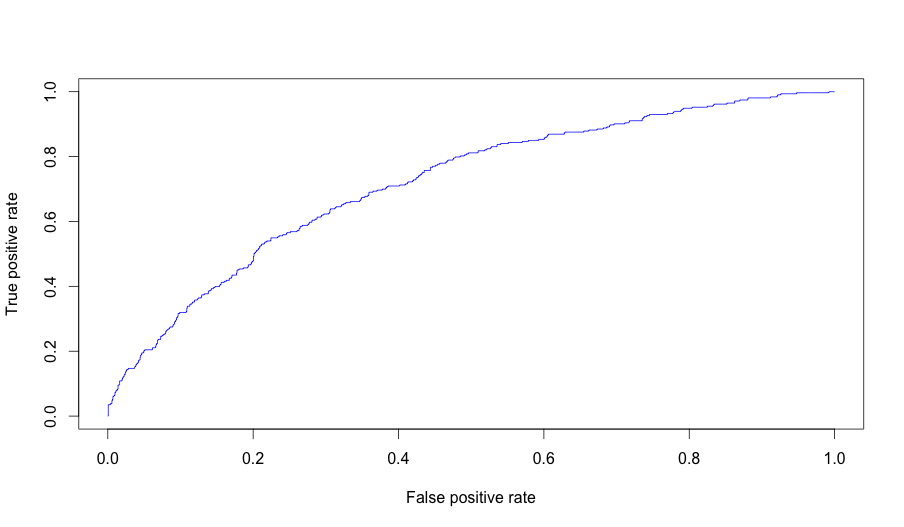
0 983 114

1 434 199

> recall

[1] 0.6357827

ROC Curve:



Performance improved over actual model as well as over Model 3. But I did not do any statistical significance test whether improvement was a significant one. I could have done it as part of k fold validation.

**Final Results on Test data:**

Now I wanted to see how my best models Model 3 and Model 4 would perform on test data. I did not train my data on train data + validation data but I used the same model which was tested on validation data. The extra data in validation data when included with training data for final model might have changed the final results because we now have more data but I doubt that it would be drastically different. Anyway, I could have tried that as well.

**Model 3.**

> AUC

[1] 0.7095134

> GC

[1] 0.4190267

> KS

[1] 0.3045184

> cutoffvalue

[1] 0.2138587

Confusion Matrix ( rows predicted, columns actual)

0 1

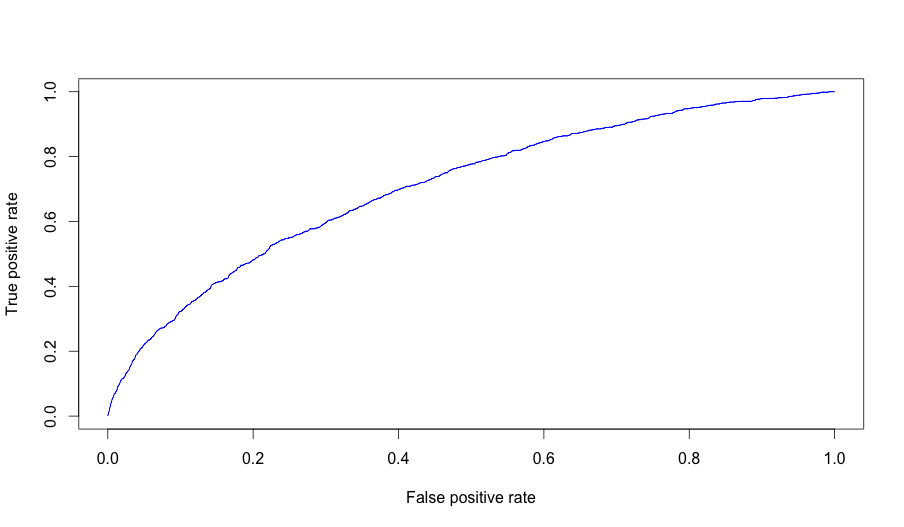
0 4623 615

1 1449 729

> recall

[1] 0.5424107

ROC Curve:



**Observation:** As you can see that AUC dropped a bit and so does the KS score. The cut-off value increased quite a lot and this caused the Recall to drop.

**Model 4:**

> AUC

[1] 0.709729

> GC

[1] 0.4194581

> KS

[1] 0.3144439

> cutoffvalue

[1] 0.1656421

Confusion Matrix ( rows predicted, columns actual)

0 1

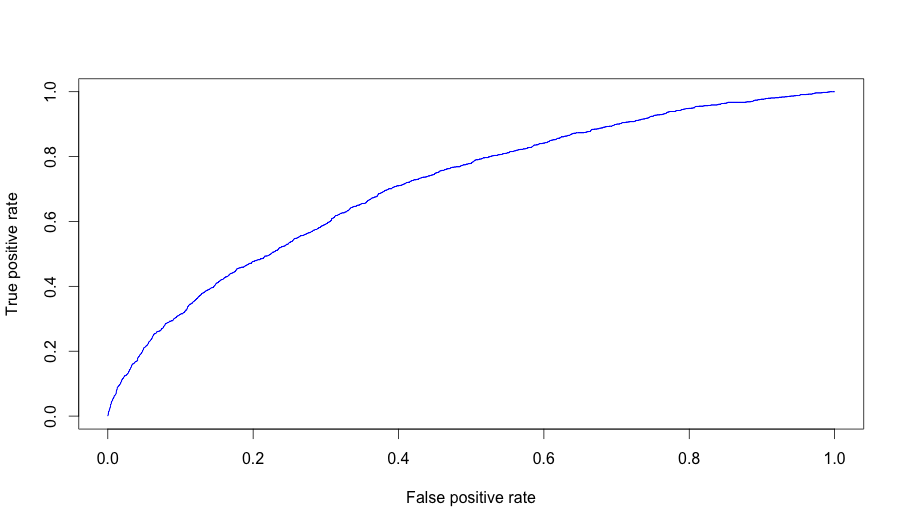
0 3730 404

1 2342 940

recall

[1] 0.6994048

ROC Curve:



**Observation:** As you can see that AUC dropped a bit and so does the KS score. The cut-off value decreased a bit and thus the recall increases.

**Note:**

Also, if you see that I merged some levels for factor columns in Model 1, 3 and 4. I tried to use the actual data without any merging but during split there were issues as some of the levels were not present in train data but were present in test and split data and thus Logistic Regression did not worked during prediction. However, at times this was not an issue and I tested the performance of that model again Model 1 on same set of data (I merged before running the Model 1). The AUC for actual data without any merging was higher by value of 0.006.

**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. I have not included the model or its results here.

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 sorted based on the coefficient values but with p values say less than 0.1 or 0.5.
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 or could have used a different sort of method to tackle the class imbalance problem. E.g. I could have created a couple of models where I could have randomly sampled the data from negative class (say 2000 rows) and then randomly sample similar number of rows from positive class. Then, I could have had say 5-10 such models and would have averaged them in the end for final predictions.

**Code:**

All my code is in github directory. 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.

I have loaded the train, test and valid data csv files to github as well so that these can be reproduced.

**Required libraries are**

Rpart

randomForest

ROCR

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

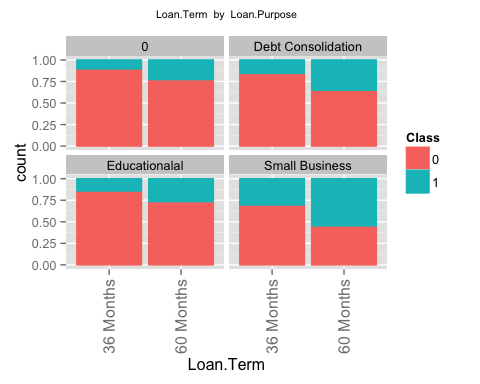
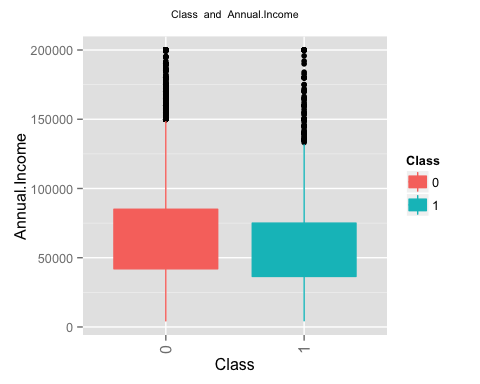
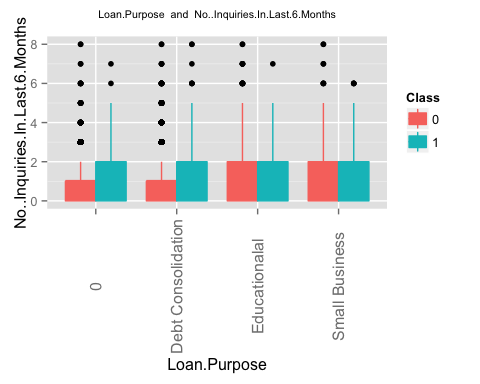
discretization

ggplot2

**The code path is**

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

Graphs to support some of interaction terms:

1. LoanAmount:Loan.Purpose. Look at small business and 60 months. Other are also showing some sort of trends. However, I do not have how many rows for small business.
2. 
3. Annual.Income:
4. 
5. Loan.Purpose:No..Delinquencies.In.Last.2.Years
6. Loan.Purpose:Use.Of.Credit.Line