UNIVERSITÉ Concordia UNIVERSITY

Computers Science and Software Engineering Department

SOEN691 – Big Data Analytics

(Winter - 2018)

Project Report By: **Kushal Kanungo**

Student IDs: 40021718

Emails: kushalkanungo1991@gmail.com

Course Coordinator: Tristan Glatard

Abstract

The project is about solving a business problem for Lending Club, which is an online peer to peer credit marketplace which matches borrowers and investors. Lending Club primarily depends on borrowers credit-history and rate their credit worthiness accordingly. This rating information is then available to investors who fund the loan requests, so that the investors can decide which loan request and how much of that loan request they will fund. This project will help the investors in deciding the credit-worthiness of their borrowers using historic loan data.

Introduction

The Business Problem:

Lending Club is an online peer to peer credit marketplace which matches borrowers and investors. For evaluating the credit-worthiness of their borrowers, Lending Club primarily relies on a grade and sub-grade it assigns them based on their credit-history. This rating information is then made available to investors who fund the loan requests, so that the investors can decide which loan request and how much of that loan request they will fund. In addition to the grade information, Lending Club provides historical loan performance data to investors for more comprehensive analysis. Our business problem is to find a model that will utilize the historic loan data to help better identify borrowers that are likely to default. Such a model would allow investors to avoid loan defaults thus limiting the risk of their investments.

The Objective:

The objective of this project is to play with the data provided by Lending Club, conduct a set of exploratory analysis and try to apply various machine learning techniques namely Decision Tree with ensemble methods namely Random Forest and Adaptive Boosting to predict weather a loan is safe or risky and compare between the two ensemble methods.

The Dataset

The dataset for this project can be found here. The dataset I took for my project is from year 2017 from quarter 1 to quarter 4. The data comes with a data dictionary describing the meaning of each column in the data. Each row in the data contains loan information about a borrower.

Collecting the data:

The data for each quarter of 2017 is available separately in csv format. After downloading the data I merged them into a single file. The merged file has 432750 rows and 145 columns. The dependent variable we are interested in predicting is *"loan_status"*. A summary of the predicted variable is given below.

Loan Status	*Count*
Current	374892
Fully Paid	39992
Late (31-120 days)	6396
Charged Off	5861
In Grace Period	4007
Late (16-30 days)	1592
Default	10

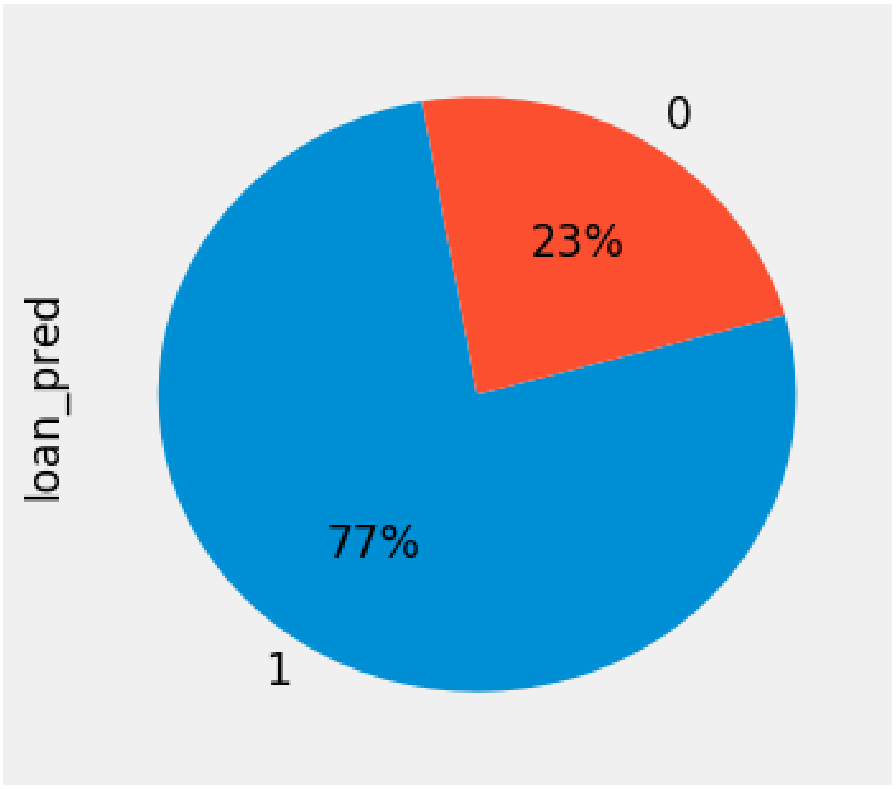
The predicted variable has 7 categories. I decided to make two classes for our predicted variable where, the category *"Fully Paid"* is considered **safe loan** and categories *"Late (31-120 days)", "Charged Off" and "Default"* is considered **risky loan**. This kind of analysis requires only historic data. One may ask, *"Default"* and *"Late (31-120 days)"* are current information about a borrower and not historic but, even if they are current data, these kind of borrows look risky as indicated by the category. The categories that are not considered are:

1. ***Current***: It means the Loan is up to date on all outstanding payments. We don't know the outcome of these borrowers.

2. ***In Grace Period***: Same as above. We don't know the outcome yet.

3. **Late (16-30 days)**: 16-30 days is acceptable.

The selected data now has 52259 rows and 145 columns. Safe loan class is assigned value 1 and risky loan is assigned value 0.



The distribution of classes in the final data ***[****<u>we can see the data is bit unbalanced but not high enough to apply SMOTE</u>****]***

The Model

The predicted variable has two classes. So, we have a classification problem. There are various Machine Learning models that are available for solving classification problems.

Classifier-(Decision Tree):

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. The following are the reasons for choosing decision tree.

1. They can capture non-linear relationship.
2. They are easy to use.
3. They are interpretable.

Ensemble methods:

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. For comparative study I chose two ensemble methods.

- **a.** ***Random Forest:*** Random forests or random decision forests are an ensemble learning method for classification, regression.
- **b.** ***AdaBoost:*** Adaboost short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gödel Prize for their work.

Packages

1. ***Scikit-learn***: A Machine Learning package in python used to train models.
2. ***Pandas:*** Pandas used for data manipulation.
3. ***Pyplot:*** Pyplot was used to plot.

Feature Selection

The data contains both categorical and continuous features. The features selection is done based on various rules:

1. Features whose description in the data dictionary are not clear are dropped.

2. Useless feature such as description, user_id etc. are dropped.

3. Features with more than 25% missing data are dropped based on condition.

   Example:

   "dti_joint" (debt to income ratio for joint application)has more than 50% missing data but it was joined with *"dti"* (debt to income ratio for individual applications) with maximum of both features as new feature. This join makes sense as we have *"application_type"* feature in the dataset. **[*More details can be found in the code base*]**

4. Features with less than 25% missing data are treated by replacing missing values with the mean if the feature is continuous and by mode if the feature is categorical.

5. Features with more than 25 categroies has been dropped.

   Example:

   "*emp_title*" has 21584 categories. To pass data to scikit-learn ensemble methods with decision tree as classifier, categories should be binary. This is because scikit-learn ensemble methods treats every variable as continuous and since a binary categorical variable has only 0 and 1, it can be treated as continuous variable that contain only 0 and 1 ***[**https://stackoverflow.com/questions/25287466/binning-of-continuous-variables-in-sklearn-ensemble-and-trees**]***. So, if our data has a categorical variable with 3 categories, then we need to do 1-hot encoding to create three binary category variables. So, for categorical variable with 25 categories we will create 25 new varaiables and in our case, for *"emp_title"* will create 21584 variables resulting in overfitting as we will end up with too much variables and less data.

After cleaning the data we end up with 59 continuous variables. (Details on features can be found in lending_club.ipynb).

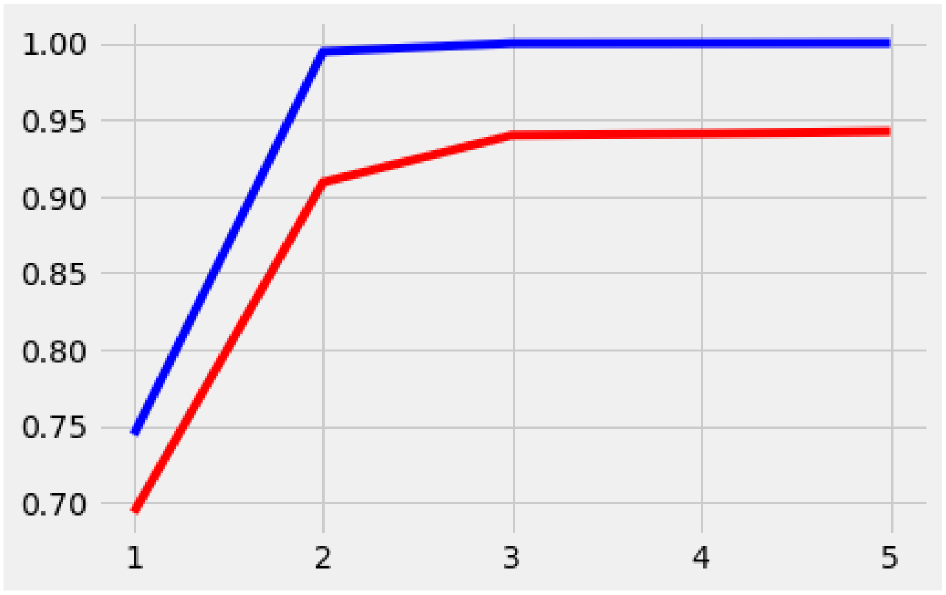
Training Model

The data is divided into test and training set with 20% in test set and 80% in training set. 10 – Fold CV is performed on the training set to select model. The metrics used to evaluate are specificity, sensitivity and accuracy.

Training the model:

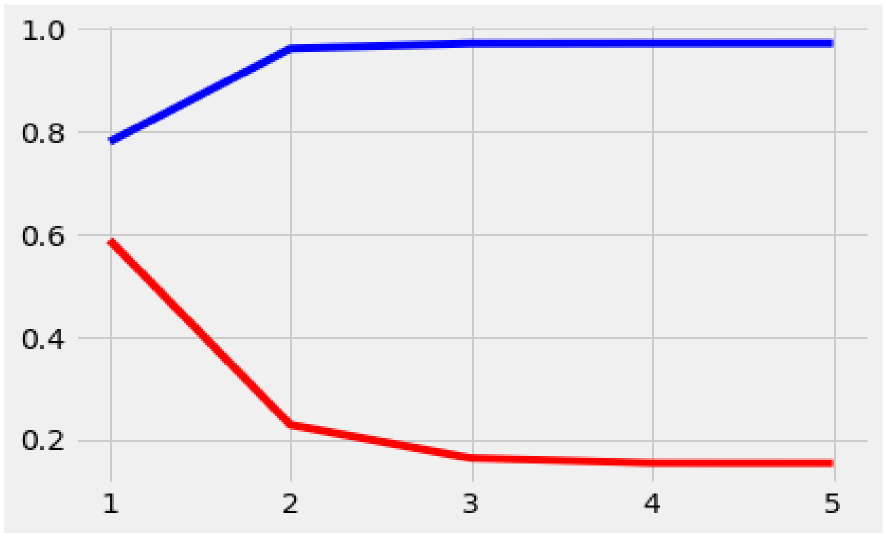
1. ***Random Forest:***

   The Random Forest model was run with 15 estimators and tree depths 10,20,30,40,50. We saw earlier that our class distribution is 23:77. Scikit-learn random forest classifier has options for class-weights. The class weight option is helpful for training model with unbalanced class. I used .23 as weight for safe loan and .77 for risky loan. Entropy is used as criteria for splitting the data by feature. The average scores on 10 fold cross validation results are given below.



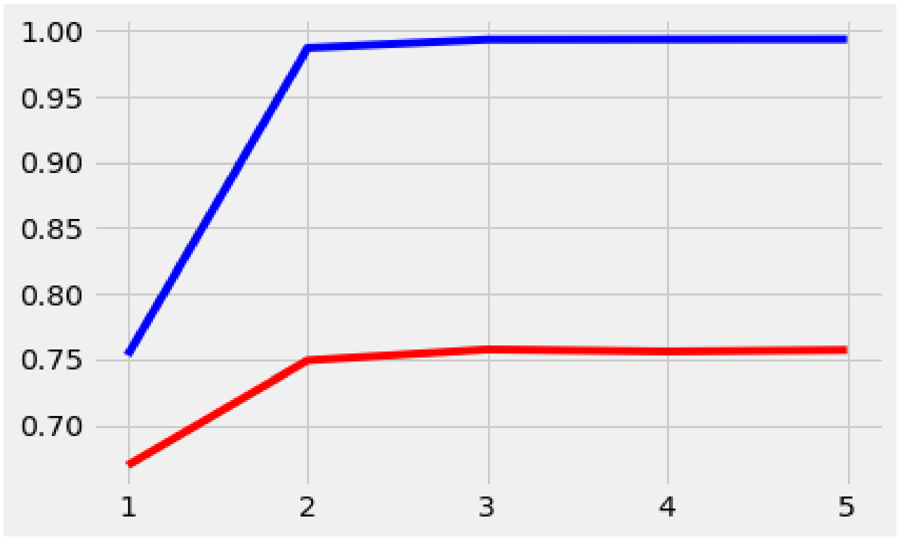
Specificity*(Recall for Safe Loan)* **[Blue:** Training Scores**; Red:** Validation Scores**]**

(x-axis : depth of tree in tens; y-axis: specificity)



Sensitivity*(Recall for Risky Loan)* **[Blue:** Training Scores**; Red:** Validation Scores**]**

(*x-axis : depth of tree in tens; y-axis: sensitivity)*



Accuracy [Blue: Training Scores**; Red:** Validation Scores**]**

(*x-axis : depth of tree in tens; y-axis: accuracy)*

After running 10 – Fold cross validation it is clear that our random forest overfits. I say this after looking at the sensitivity score. After tree depth 20, the training scores are almost 1 a clear sign of overfitting. From tree depth 10 to 20 sensitivity drops rapidly. Detecting bad loans is important, so I choose random forest model with tree depth as 10 since it has the best sensitivity score.

Final Model:

The final model was run on training data and tested on test data with tree depth as 10. The test results are given below.

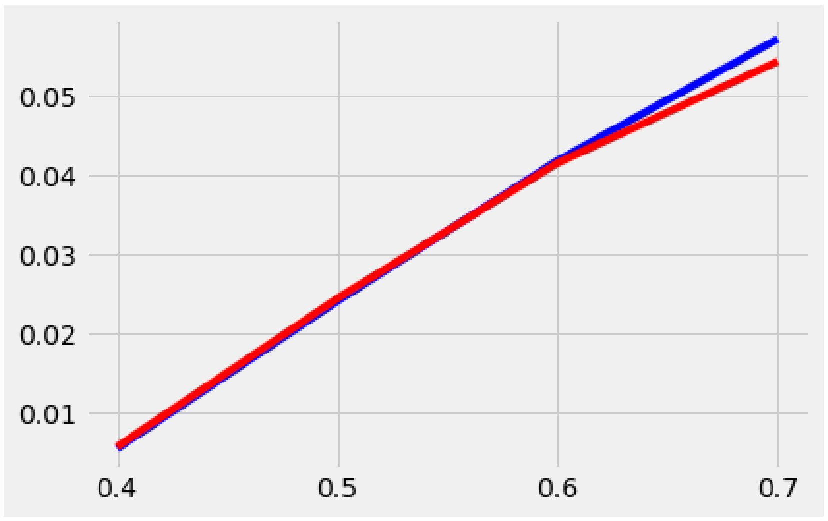
Specificity	0.819227403425
Sensitivity	0.381573583367
Accuracy	0.716513585917

The confusion Matrix is given below.

TP:*6553*	**FN:** *1446*
FP: *1517*	**TN:** *936*

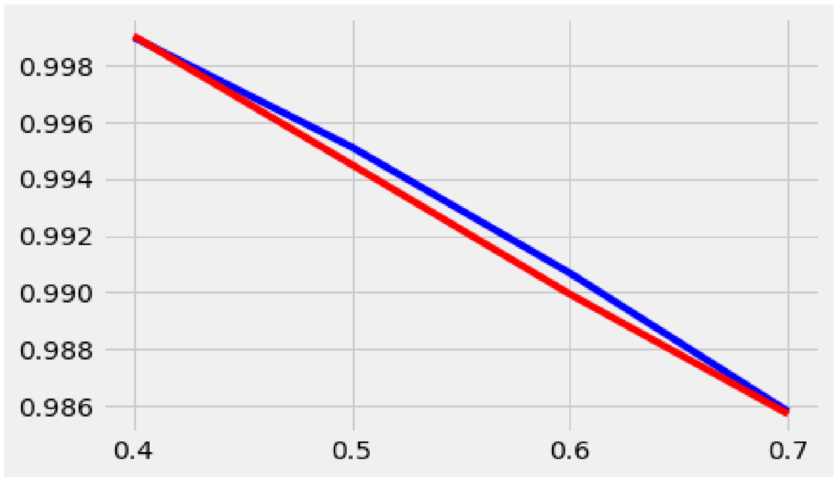
2. ***Adaboost:***

The Adaboost model was run with 15 estimators and with learning rates 0.4, 0.5, 0.6, 0.7. Unfortunately scikit-learn Adaboost classifier has no class weight option. The average scores on 10 fold cross validation results are given below.



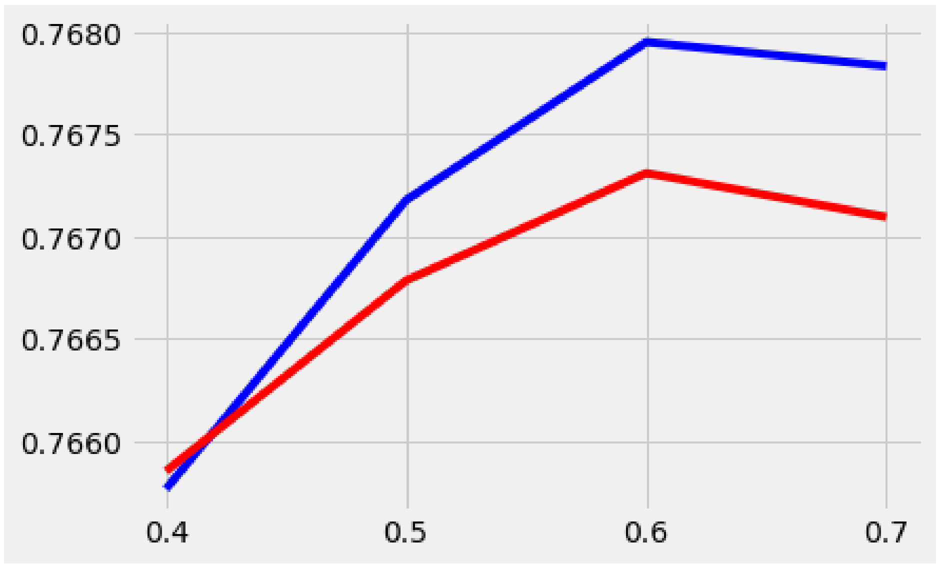
Sensitivity*(Recall for Risky Loan)* **[Blue:** Training Scores**; Red:** Validation Scores**]**

(x-axis learning-rate; y-axis: sensitivity)



Specificity*(Recall for Safe Loan)* **[Blue:** Training Scores**; Red:** Validation Scores**]**

(x-axis : learning-rate; y-axis: specificity)



Accuracy [Blue: Training Scores**; Red:** Validation Scores**]**

(x-axis : learning rate; y-axis: accuracy)

Final Model:

The final model was run on training data and tested on test data with learning rate as 0.6 since validation accuracy decreased after that. The test results are given below.

Specificity	0.990498812352
Sensitivity	0.0501426824297
Accuracy	0.769804822044

The confusion Matrix is given below.

TP: *7923*	**FN:** *76*
FP: *2330*	**TN:** *123*

Conclusion

It would be unfair to make a direct comparison between the two models used here because, one of them had the option to include class weights. But if I am to choose, despite having more accuracy than Random Forest, I would not choose AdaBoost model because the sensitivity scores are too bad. Random Forest has decent accuracy with much higher sensitivity score. So. I would declare it as the winner.

[Code Base] : https://github.com/KAY-YAK/Big-Data-SOEN-691