Predicting Loan Defaulting

Robert Kinini rkinini@stanford.edu
Kenneth Nturibi knturibi@stanford.edu
Prinslou Tare prinslou@stanford.edu
Department of Computer Science
Stanford University

Abstract

This project aims to predict the likelihood of people defaulting on loans. In this project milestone we implemented various algorithms on our data such as Random Forrest, Logistic Regression and Multi-Layer Perceptron Classifier.

1 Motivation and Problem Statement

Following the US financial crisis of 2008, credit institutions faced an explosive number of loan defaults. According to the Federal Statistics Office, 2013, the number of newly opened insolvency proceedings rose from about 95,235(2004 to 2012) to 134,250 in 2013. This jeopardized the profitability of banks due to the great number of nonperforming loans. Following these developments, there is a greater need for an early detection of risks of default in the world of finance. The main challenge revolves around creating models that not only have high detection rates but also deliver the detection at an early stage. This will in effect buy more time for the credit institutions to take effective measures to handle non performing loans and eventually prevent losses.

1.1 Similar Projects and Literature

We have been able to read up on various papers that tackled the issue of loan review using Machine Learning techniques. From these readings, we have gained a better understanding of how to implement machine learning models to suit or goals. We also intend to use some of the models implemented in these papers as a benchmark. Here are some papers that we have been able to examine so far: (1) Loan Approval Prediction based on Machine Learning Approach by Kumar Arun, Garg Ishan and Kaur Sanmeet (2) A machine learning approach for predicting bank credit worthiness by Regina Esi Turkson ; Edward Yeallakuor Baagyere ; Gideon Evans Wenya (3) Developing Prediction Model of Loan Risk in Banks Using Data Mining by Aboobyda Jafar Hamid and Tarig Mohammed Ahmed. (4) Credit Risk Prediction Using Artificial Neural Network Algorithm by Shruti Goyal.

2 Description of the Data

We used loan data publicly available from the Lending club. Given that the Lending club has a very huge data bank, we chose to use data from 2007 to the first quarter of 2016. We felt that this amount of data this would be sufficient to meet our original intended goals. The rows in the dataset were the different examples and the columns were the features. The feature $loan\_status$ indicated whether one had defaulted on a loan or not. It is important to note that the data had skewed classes, with 660,000 negative examples and 150,000 positive examples.

CS230 Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Data Preprocessing

After collecting the data, we had to deal with the challenge of preprocessing it. We did this in various stages. First, given that our main interest at the moment was to predict whether a person would default on a loan or not, we decided that we would only retain data with a relevant loan status. Hence, we ended up only retaining data that had a loan status of "Fully Paid" or "Charged off". This left us with 818501 examples in the dataset, each with 132 features. We then got rid of features that had more than 60% of their entries being null. This brought us down to 88 features from 132. After doing so we had to decide on the set of features that were most relevant to our prediction task hence we chose the 23 features that we used in our notebook. We then converted all categorical variables into numeric which included binarizing some features and encoding others. We set the 'loan_status' feature as target_variable labeling 'Fully_paid' status as "1" and 'charged off' status as "0".

3.1 Feature Hashing vs One Hot Encoding for Categorical Features

It was difficult to encode some features such as the *zipcode* and the *state*. Since using both features was redundant, we used the more specific one: *zipcode*. Due to privacy reasons, the Lending Club had anonyomized this data altering the last three digits. For instance, Stanford's zip code: 94305, was coded as $94xxx$. We noted that the zipcode was a crucial indicator if a client would default on a loan so we knew that it was important to encode it to improve the performance of our models from the milestone. We therefore tried the following encoding schemes for the zipcode:
Feature Hashing
In this scheme, a hash function is used to map a large number of values to a small finite set of values. In our case, we hashed the zip codes in a smaller set of finite integer values and fed these values to our model.
One Hot Encoding
If we have m labels, the one hot encoding scheme transforms each attribute into m binary features where the label corresponding to the attribute is encoded as 1 and the rest are zeros. This scheme has the downside of making training take longer than expected due to the sparsity of the one hot vectors.
Splitting Dataset
We finally did feature scaling on the data frame and then split the dataset on a 70/30 basis for training and testing respectively.

4 Modelling

After preprocessing our data, we then did a 70-30 train-test split on the data and attempted a number of different machine learning models in order to map the baseline accuracy of our problem. We noted no significant change in performance when we used the one hot encoding scheme and therefore, in this section we will compare the performance of our models with feature hashing versus without feature hashing. We hashed the zipcodes to generate 6 features

4.1 Logistic Regression Model (Baseline)

We implemented logistic regression model as our baseline model and got an accuracy of 67.3%. We then came up with a classification report as follows:

Class	Precision	Recall	F1-score	Support
0	0.89	0.68	0.77	133632
1	0.31	0.64	0.42	30269

4.2 Random Forest

In addition to implementing Logistic regression as part of our baseline, we also implemented the Random Forest model. We chose this model based on previous works in this domain and advice received at office hours. The Random forest approach obtained an accuracy of 81.8%

Class	Precision	Recall	F1-score	Support
0	0.83	0.98	0.90	200156
1	0.55	0.08	0.15	45695

4.3 Multi-layer Perceptron

The last implementation for our baseline was the Multi-layer perceptron. We used this for the purpose of regression analysis. Using this approach was inspired by the paper *Neural Network Approach to Loan Default Prediction* and advice from our project mentor. In implementing this model, we used a reLu activation and used adam optimization. This approach helped us obtain an accuracy of 81.8%. The classification report is shown below.

Class	Precision	Recall	F1-score	Support
0	0.82	0.99	0.90	200156
1	0.56	0.07	0.12	45695

5 Hyper Parameter Tuning and Advanced Algorithms

Driven by the need to get better results, we analyzed the different feature importances and implemented boost algorithms with the aim of improving our performance

5.1 Feature Importance

Using the Random Forest Model, we were able to compute the importance of the various features in our dataset. The features shown as numbers(0, 1, 2, 3...) are as a result of feature hashing for zip codes and states. We started off with many columns but eventually narrowed on the features plotted below with their importances.

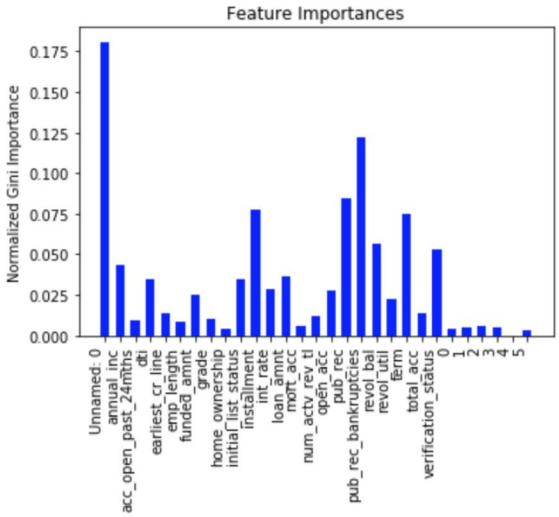

Figure 1: Feature Importance

5.2 XGB Boost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. We implemented this model since we felt that the implementation's laser focus on computational speed and model performance would be helpful in increasing our model's accuracy. Moreover,the model is very useful for predictive modeling in regression and classification, especially when dealing with structured or tabular datasets.The XGB Boost model obtained an accuracy of 81.8% .

Class	Precision	Recall	F1-score	Support
0	0.83	0.99	0.90	133632
1	0.58	0.08	0.12	30269

5.3 Light GBM Boost

Another model we implemented was Light GBM Boost. LightGBM is a gradient boosting framework that uses tree based learning algorithms. We decided to implement this model given the fact that it presents us with the following advantages;faster training speed and higher efficiency,lower memory usage,better accuracy,support of parallel and GPU learning and its capability to handle large-scale data.For this model, we got the following results. The LightGBM model obtained an accuracy of 81.9%.

HyperParameter Tuning

We adjusted the different hyperparameters for this model and got our optimal parameeters to be : number of leaves : 200, learning rate: 0.001, bagging fraction: 0.9 and the maximum depth to be 20. Using these parameters, we got a classification report as follows:

Class	Precision	Recall	F1-score	Support
0	0.83	0.98	0.90	133632
1	0.54	0.11	0.19	30269

6 Evaluation Metric and Result Analysis

For the logistic regression classifier, cross entropy loss is used and it is characterized by the following cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right].$$

where $\left(h_\theta\left(x^{(i)}\right)\right.$ is the predicted value and $y^{(i)}$ is the actual value. The Multi-layer Perceptron makes use of log loss which is similar to cross-entropy loss when it comes to binary classification.

As we can see from the model performance, it is evident that even though the Random Forrest and MLP models do have a high overall accuracy and high precision and recall for the "0" class("Fully paid"), they have very low precision and recall when it comes to the "1" class("charged off" class). The precision of the "1" class is really poor with around a 57% maximum among the models. The recall of the "1" class is much worse with a maximum of a 0.08 percent recall among the models. This shows that more than 90% of the charged off debts are being mis-classified by the models. Though the logistic regression classifier has a lower accuracy than the other two models, it has better precision and recall for the charged off class.

Ultimately, the results show us that the models we used for our baseline are insufficient in terms of prediction. Hence, we need to try different models that can make sense of the complexities in the data so as to get better accuracy.

7 Challenges

The lending club data has a lot of empty or sparse columns and rows which needed intensive prepossessing. We choose to discard all columns which were more than 60 percent empty.

Some of the features were partly anonymity, e.g zip code, which limited the full potential of encoding such a feature into the train data set and evaluate the effect it has on performance.

On the ethics side, there is the issue of bias. As much as we try to create a fair model in terms of the features we choose to use, our model can just be as good as the data. If the data we have is biased towards people of a given demographic(from zipcode encoding), our model will just enhance the same.

Lastly, in real life, if a deep learning model predicts that a client is most likely to default on a loan, the bank will take stern actions such as deny extending the loan period or deny topping up the loan of

a business/person. Such a decision has serious life altering effects such as shutting down of a business or inability to get cash for cancer treatment.It is very important that such decisions are accompanied by a detailed explanation. However, we cannot fully understand or explain to a client why a deep learning model decided that they are likely to default on a loan. Enforcing such decisions in real life turn to be a challenge because Deep learning models are unable to provide explanations for their predictions.

8 Future Work

We encountered relatively low precision and recall on the class 1. This can be attributed to the fact that our data was skewed. In fact, the skewdness was at 1.6. Therefore, most of our future work will be focused on fixing this skewdness. Here are some of the steps we plan to take:
UnderSampling
We plan to under-sample negative examples so that we end up with an equal number of positive and negative examples.
Collect or Synthesize More Data
Since under-sampling may result in fewer training examples overall, another approach that can be taken is collecting more data for the positive examples so that we have an almost equal representation of positive and negative classes in our dataset.

References

[1] S. Goyal, "Credit Risk Prediction Using Artificial Neural Network Algorithm," Data Science Central. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/credit-risk-prediction-using-artificial-neural-network-algorithm. [Accessed: 20-Feb-2019].

[2] "Personal Loans Borrow up to $40,000 and get a low, fixed rate.," Peer to Peer Lending Alternative Investing. [Online]. Available: https://www.lendingclub.com/info/download-data.action. [Accessed: 20-Feb-2019]. CoRR abs/1401.0864, 2014.

[3] otwani, Anand Chaurasiya, P Bajaj, G. (2018). Predicting Credit Worthiness of Bank Customer with Machine Learning Over Cloud. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING. 6. 1471-1477. 10.26438/ijcse/v6i7.14711477.

[4] https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/