

Capstone Project

Machine Learning Nanodegree

Rodney Sales Nogueira Jr

November 2017

1 Definition

1.1 Project Overview

In this project, I created machine learning models to analyse credit scoring based on the Kaggle Give Me Some Credit Competition [12]. This study is focused on our company ongoing *AI project* to analyse potential credit frauds, and credit scores.

”The need for credit analysis was born in the beginning of commerce in conjunction with the borrowing and lending of money, and the purchasing authorization to pay any debt in future. However, the modern concepts and ideas of credit scoring analysis emerged about 70 years ago with Durand” [10].

A credit score is a numerical expression based on a level analysis of a person’s credit files, to represent the creditworthiness of an individual. A credit score is primarily based on a credit report information typically sourced from credit bureaus [1]

So the main idea of credit scoring models is to identify the features that influence the payment or the non-payment behavior of the customer as well as his default risk, occurring as the classification into two distinct groups characterized by the decision on the acceptance or rejection of the credit application [10].

Some of the problems are to solve a common supervised learning approach to credit scoring, as the scores can be a numerical expression in a certain range, such as a FICO Score: 300-850. Our particular goal is to comprehend how the machine learning design process works for applications in this domain. So credit scoring is a good way to start.

The goal of this project is to understand, and learn the challenges of building a machine learning model to analyse credit scoring, based on publicly available data. The idea is to have a overview and a start in the solutions for kind of problems.

1.2 Problem Statement

As stated by [10] sometimes the decision is binary based on an acceptance/rejection rate, meaning that this problem can be treated as a classification problem. West [2] defines that the objective of credit scoring models is to assign credit applicants to two groups of credit: bad credit group, good credit group. The first model applied to these problems were linear discriminant analysis [2].

The data used in this kinds of problem have common features such as: age, credit status, expenses, income, assets, marital status, etc. The credit industry grows substantially everyday, people tend to neglect their payments, the industry is developing models to prevent frauds, and to evaluate how much credit a person might need. Even a fraction of a percent increase in this kind of problems is a significant accomplishment [2].

There are a lot of machine learning models being applied to this kind of problems, neural networks [2], support vector machines [4], logistic regressions, discriminant analysis, decision trees, k-nearest neighbours [8], among others.

Our problem was focused on the Kaggle Give Me Some Credit Competition to study the data, and focus on selecting a good set of features, and creating a Model to a late submission on the competition. We focus on a supervised learning solution using ensemble models, which are already showing good results for this problem [7, ?], and deep neural networks that also showed excellent results, having the best accuracy on a australian/german dataset [?].

1.3 Metrics

There are several performance measures to the field of credit scoring, such as kullback divergence, the H measure, the Receiver Operating Characteristic, and the area under the receiver operating characteristic curve (AUC) is one of the most commonly used measures for evaluating predictive performance. The AUC is used to measure a predictive accuracy to evaluate different machine learning models.

Most of the credit scoring cases denote a binary problem, regarding where the customer is a good, or bad customer. The nature of some classifiers is to predict a probability value, between 1 and 0, a common accuracy metric will compute by a threshold, like 0.5, and if the value is below that threshold would be considered 0, or above 1. Due to this nature of the problem, the AUC is able to determine multiple thresholds based on the actual data, and measure more accurately the binary decision problems, meaning that a 0.80 value would have a different score than a 0.90 value for a positive (1) prediction.

Since our problem and goal is part of a Kaggle Challenge, the selection of the AUC measurement is also a pre-requisite for the competition [12].

2 Analysis

2.1 Data Exploration

The Give Me Some Credit dataset contains both a training set, and a testing set. The training set contains 150000 entries, and the test set contains 101503 entries, but the test set does not contains the *ground truth* variable.

The features of the dataset are:

- SeriousDlqin2yrs, yes or no, and it's the class value
- RevolvingUtilizationOfUnsecuredLines
- Age
- NumberOfTime30-59DaysPastDueNotWorse
- DebtRatio
- MonthlyIncome
- NumberOfOpenCreditLinesAndLoans
- NumberOfTimes90DaysLate
- NumberRealEstateLoansOrLines
- NumberOfTime60-89DaysPastDueNotWorse
- NumberOfDependents

The dataset label is the SeriousDlqin2yrs, with two classes, yes if a person experienced 90 days past due delinquency or worse. The features includes two percentage values (RevolvingUtilizationOfUnsecuredLines, DebtRatio), a real value (MonthlyIncome), and the rest are integers values. The mutually dependent variables might be the monthly income, and the debtratio, since the person debts are always linked with the income, a person with a higher income will have a lower debt ratio.

Both classes for past due (NumberOfTime30-59DaysPastDueNotWorse, and NumberOfTime60-89DaysPastDueNotWorse) are also really important, since they are related to how many times the person experienced financial problems. The table1 shows us a sample from the data, we can see that there are a great discrepancy between the variable values, from values like 63588.0, against small values like 0.121876, this states that the dataset clearly need a data preprocessing, a scaling, or some more complex form of pre processing.

The exploration phase also included searching the competition forum, looking for the competitors thoughts on the data. As stated there is a problem with the class imbalance of the data.

| SeriousDlqin2yrs | RevolvingUtilizationOfUnsecuredLines | age | NumberOfTime3059DaysPastDueNotWor |
|------------------|--------------------------------------|-----|-----------------------------------|
| 1 | 0.766127 | 45 | 2 |
| 0 | 0.957151 | 40 | 0 |
| 0 | 0.658180 | 38 | 1 |
| 0 | 0.233810 | 30 | 0 |
| 0 | 0.907239 | 49 | 1 |

Table 1: Data sample

2.2 Data Visualization

The training set contains 150.000 entries, the frequency is 93.3% for the persons who are not experiencing financial distress (Class 0), and 6.7% for the persons who are experiencing financial distress (Class 1), as shown in the Figure 1. This shows us a great problem and a challenge, there is a big class imbalance in the data, and metrics such as accuracy might not be a good way to validate this kind of data.

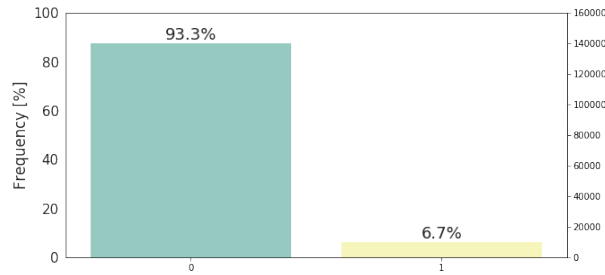


Figure 1: Dataset Class Frequency

More analysis of the data ¹ shows some interesting results, such as the percentage of the missing (NaN) values, which in the column of the MonthlyIncome there is a 19% of missing values.

Also, we can see that the number of occurrences for the people ages grows on the dataset, there are more older people on the data. And we also cross checked that older people tend to have more financial problems.

The same exploration was done with the DebtRatio and the Monthly income, and we can analyse that lower MonthlyIncome tend to have a higher DebtRatio, which makes sense.

2.3 Algorithms and Techniques

According to the forums [12, 15, 16] the algorithms which achieved good results were:

¹<https://github.com/rodsnrj/credit-scoring/blob/master/Data.ipynb>

1. Random Forest
2. Gradient Boosting (XGB)
3. QDA Ensemble
4. Neural Networks

As planned in the capstone I selected simpler models to analyse the data, and followed with the models used in the forums. The simple models selected were: gaussian naive bayes; logistic regression; and decision trees.

After the initial analysis I narrowed my choices to two approaches: a deep fully connected multi layer perceptron neural network, which is a trending nowadays with all the recent state of the art deep neural networks; and the gradient boosting classifier which shown good results in the competition leaderboards, and my initial tests.

An artificial neural network is a biologically inspired algorithm, that perform certain tasks, like clustering, classification, or regression. They can be viewed as weighted directed graphs, where neurons are nodes, and directed edges with weights are the connections between the neuron outputs and inputs, as shown in Figure 2. The input layer is feed with the features of our data, the hidden layers are the networks core, where it learns to classify our data, and the output layer give us the final result, depending on the network architecture. For a binary classification network, the output layer would give us a percentual value based on the current classes, closer to 0, or 1 [11].

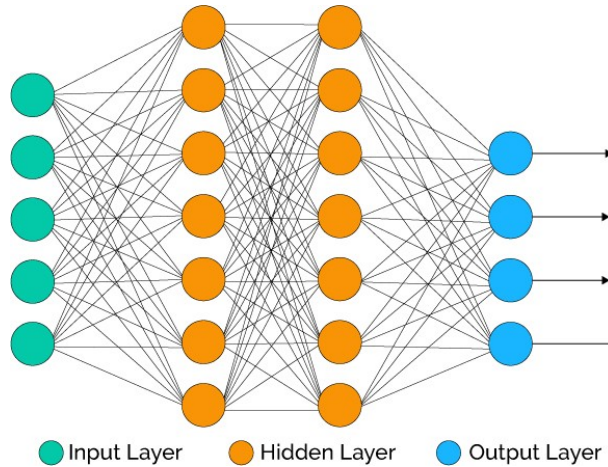


Figure 2: Artificial neural network overview

The Figure 3 shows an overview of the operations of the network neurons, or layers. The layer operation is a matrix multiplication followed by a activation function of the neuron, and optionally the addition of a bias, denoted by: $f(X * W) + b$.

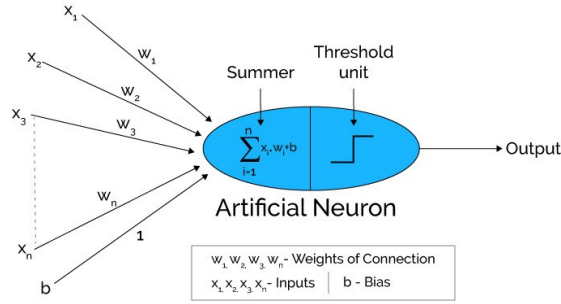


Figure 3: Artificial neural network neuron

There are several neural network architectures, as shown in Figure 4. For our study we use a simple fully connected multi layer perceptron model. The training this neural networks is based on the gradient descent, and back propagation algorithm. In back propagation the error (different between the desired output, and current output), is propagated backward from the output layer to the input layer updating the weights of the hidden layers. This process is done for several iterations on the training data [11].

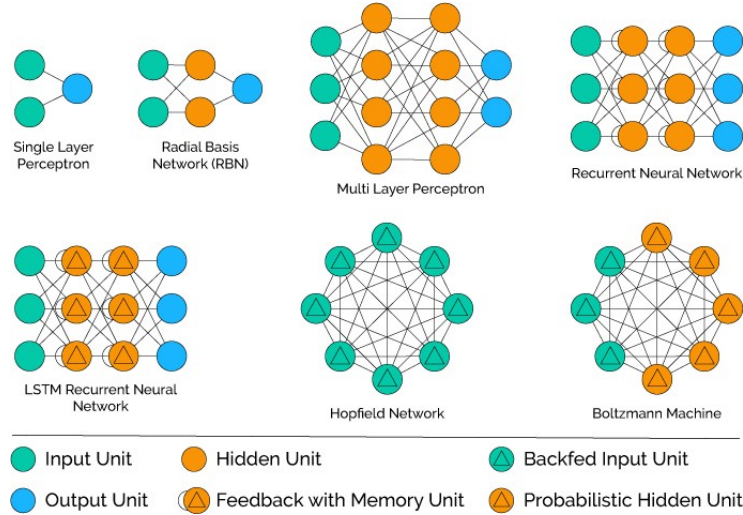


Figure 4: Popular neural network architectures

The power of a deep neural network is that each hidden layer is able to describe and learn how features are connected, and each layer is able to built a non linear classifier over the data. Also the deep networks can have several types of configurations and architectures, this makes them extremely versatile, but it's also time consuming to find a good architecture for the given problem. Also the [?] proposed deep network achieved the best accuracy for their given

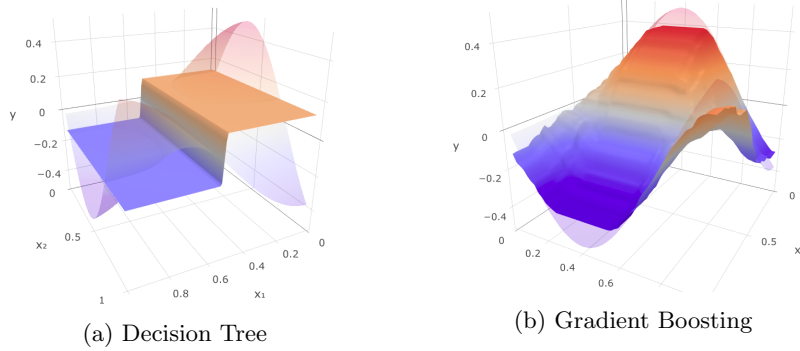


Figure 5: Decision Tree vs Gradient Boosting

dataset.

And a Gradient boosting algorithm is an ensembling technique, which means that prediction is done by an ensemble of simpler estimators. While this theoretical framework makes it possible to create an ensemble of various estimators, in practice we almost always use GBDT — gradient boosting over decision trees. This is the case I cover in the demo, but in principle any other estimator could be used instead of a decision tree [13].

The aim of gradient boosting is to create (or "train") an ensemble of trees, given that we know how to train a single decision tree. This technique is called boosting because we expect an ensemble to work much better than a single estimator [13].

As shown in Figure 5, a single decision tree (14a) can build a linear classifier over the data, a combination(ensemble) of several decision trees (14b) can powerfully fit the model to the data. This is a good option for non linear complex datasets. And also a Gradient Boosting model [16] achieved one of the best results in the [12] given challenge, also there are several gradient boosting approaches optimized with the AUC score as presented by Kraus [3].

The dataset is also unbalanced, as shown in the data exploration, according to [6] ensembling techniques are really useful, but gradient boosting have some disadvantages: they are harder to fit than their counterparts (random forests); and they generally have 3 parameters which can be fine-tuned, Shrinkage parameter, depth of the tree, the number of trees. Proper training of each of these parameters is needed for a good fit. If parameters are not tuned correctly it may result in over-fitting, which means they can be harder to train.

2.4 Benchmark

To evaluate my testings, and benchmark my models, I used the hiper-parameters and configurations exposed in the Competition Forums, with the models used by them, and evaluated it on a test set made with 20 to 30% of the data in the training set.

The mean AUC for a 10 fold cross-validation from all of the following were:

- GaussianNB: 0.71
- DecisionTreeClassifier: 0.61
- LogisticRegression: 0.70
- MLPClassifier: 0.83
- AdaBoostClassifier: 0.86
- RandomForestClassifier: 0.78
- GradientBoostingClassifier: 0.86
- BaggingClassifier: 0.78
- XGBClassifier: 0.86

And for the mean AUC of a 10 fold cross-validation, for the experiments done in the 3rd competitor's feature set:

- GaussianNB: 0.83
- DecisionTreeClassifier: 0.61
- LogisticRegression: 0.86
- MLPClassifier: 0.58
- AdaBoostClassifier: 0.86
- RandomForestClassifier: 0.78
- GradientBoostingClassifier: 0.86
- BaggingClassifier: 0.78
- XGBClassifier: 0.86

Then I started to develop my own tests with my configurations on the models, and compared with those results.

And lastly I used the *late submission* of the competition to benchmark the final result of each one of my selected models. I aimed for score of at least 0.86, but my goal was to get close to the sample entry benchmark (0.864248) as shown in Figure 6.


| | | | | | | |
|-----|------|--|--|----------|----|----|
| 356 | ▼ 62 | AdjustedRSquared |  | 0.864250 | 12 | 6y |
| | |  Sample Entry Benchmark |  | 0.864248 | | |
| 357 | ▼ 60 | Anthony Goldbloom |  | 0.864248 | 2 | 6y |
| 358 | ▼ 60 | bamboochen |  | 0.864248 | 1 | 6y |

Figure 6: Kaggle Competition Sample Entry Benchmark

3 Methodology

3.1 Data Preprocessing

To start the simple evaluation of the classifier I've preprocessed the training feature set with the Scikit-Learn Standard Scaler ². And uploaded a submission with the GradientBoosting, which achieved 0.77 AUC in the Cross-Validation, as shown in Figure 7 the simply scaled data is not enough to achieve a good result.

| Your most recent submission | | | | |
|--|-------------------|-----------|----------------|----------|
| Name | Submitted | Wait time | Execution time | Score |
| first_submission_gb.csv | a few seconds ago | 8 seconds | 1 seconds | 0.843094 |
| Complete | | | | |
| Jump to your position on the leaderboard ▼ | | | | |

Figure 7: Submission on the Simple Scaled Feature Set

The pre processing of any given dataset requires a lot of study, and background research for the field of research, so for the data pre processing I based myself in the code kindly shared by the 3rd place on the competition [15]. The implementation is on the Data Transform Notebook ³, there are a lot of log values which are a good way to scale the features (they are denoted by the *Log* prefix).

The dataset does not offer many distinct features, or clear class cross reference, so I also had to do that, an example is: the different debt ratio values which were distinguished with the *DTIOver33*, *DTIOver43* columns; and the cross reference between the late payment columns (*NumberOfTime3059DaysPastDueNotWorse*, *NumberOfTime6089DaysPastDueNotWorse*) with the column *NumberOfTimesPastDue*.

The final feature set consisted in 80 features varying from the Log values of the main features, cross reference between each feature, binary features to some behavior within the dataset, like the number of dependents being 0, or the

²<http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

³<https://github.com/rodsnrj/credit-scoring/blob/master/Data-Transform.ipynb>

number of people who are eligible for the social security (above 60 years), and so on. Also all the infinity, or missing features (NaN) were filled with the 0 fill technique.

3.2 Implementation

The implementation is split in three main stages:

- Implementation of the data handling, pre processing, and visualizations
- The implementation of the classifiers, including training, splitting and testing
- The implementation and refinements of the algorithms, and of the submissions generation functions

For the first stage I started with the exploration of the discussion threads from the Kaggle Competition, and based some of my implementations according to the community shared studies, and codes.

The data handling was made with Pandas, Numpy, and Scikit-Learn toolkits, using both Pandas Queries, and and Numpy utility functions. Scikit-Learn was also used to pre-process and select my features. And the visualizations were made using Matplotlib plots and the built-in plots in Pandas.

The final data pre-processing was solely based on the 3rd place shared *R* script. Where I extracted the same 80 features to compare with my own previous feature selections.

The implementations of the classifiers then followed using Scikit-Learn, in addition with Keras to build my Neural Network models. I've created a few helper functions to generate multiple results from multiple classifier trainings and scorings.

I've built simple functions to: train and test, multiple classifiers, with or without cross-validation; split the dataset into training and testing; and I've also built functions to generate the submission file from both Keras models, or Scikit-Learn classifiers;

Since the implementations were used to the Kaggle competition, which is solely based on finding a good classifier, and submitting its results. I did not focused on having a organized or performatic code.

3.3 Refinement

As stated for the final part of the project I've chosen two models: a artificial neural network; and a gradient boosting classifier.

To refine the models the training dataset was split in two parts: a training set, and a testing set with 20% of the data.

The artificial neural network model refinement included:

- Adding more hidden layers, and changing their input/output size

- Changing the activation functions
- Changing the loss function, and the optimizers.

Then followed by training with 10 to 50 epochs, using a batch size of around 2048/4096. After the training is finished the model is evaluated in the test set. The model was considered good when the AUC score was above 0.76, which was the benchmark given from our previous stages, then I would submit the model to the challenge's test set, and document the results.

For the refinement of the Gradient Boosting classifier I've used the Grid Search 10-fold cross validation with a set of combinations from the following hiper-parameters of the model:

- number of estimators, 100, 200, 300, 400, and 500
- max depth, between 3, 4, 5, 6, 7 and 8
- criterion, all of the available from scikit-learn
- min samples split, 2, 3, 4, 5, and 6
- max features, all of the available from scikit-learn
- min samples leaf, 0.6, 0.8, 1.0, 1.2, 1.5, and 2

The same was done to evaluate the model performance, first I would test it on my test set, then I would submit to the challenge's test set, and document the results. If no satisfactory results were achieved, a new set of hiper-parameters would be chosen.

I did this because of hardware limitations, both trainings would take a really long time with my hardware. The possible results would be better with a longer training time, and a bigger set of possible hiper parameters.

3.4 Model Evaluation and Validation

For the evaluation of the neural network I've used a training and a testing set. After the model achieved a somewhat plausible result I've tested the model classifying the data from the competition's available test set, generated the submission file, submitted and if the result was satisfactory I would then document it and follow with the model refinement if needed.

The first model I've trained was a four layered multi layer perceptron with only sigmoid activation functions on all of the layers, as shown in Figure 8. The Figure 9 shows us a result of 0.83 on the challenge's test set.

| Layer (type) | Output Shape | Param # |
|--------------------------|--------------|---------|
| dense_1 (Dense) | (None, 80) | 6480 |
| dense_2 (Dense) | (None, 40) | 3240 |
| dense_3 (Dense) | (None, 20) | 820 |
| dense_4 (Dense) | (None, 1) | 21 |
| Total params: 10,561 | | |
| Trainable params: 10,561 | | |
| Non-trainable params: 0 | | |

Figure 8: First Neural Network Model

| Your most recent submission | | | | |
|--|-----------|-----------|----------------|----------|
| Name | Submitted | Wait time | Execution time | Score |
| submission_last.csv | just now | 2 seconds | 1 seconds | 0.837229 |
| <div>Complete</div> | | | | |
| Jump to your position on the leaderboard | | | | |

Figure 9: First submission from the simple neural network model

The best neural network model I've come up with is shown in Figure 10.

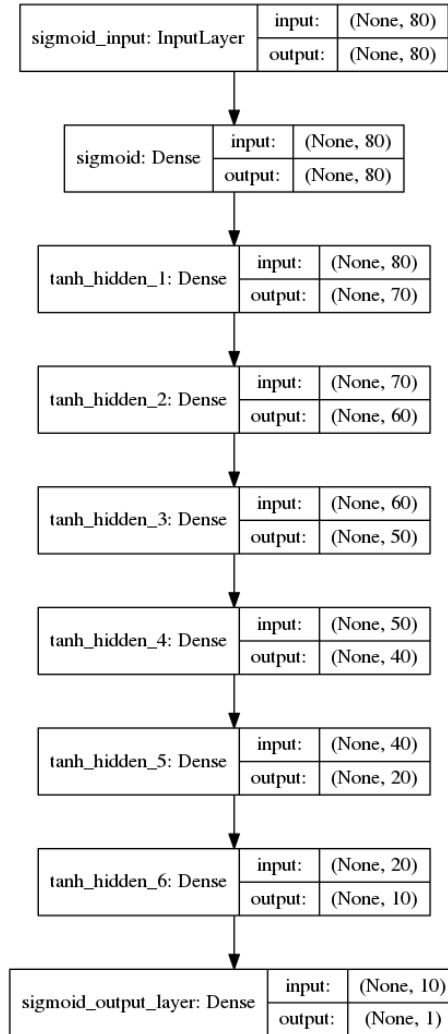


Figure 10: Multi Layer Neural Network Architecture for Credit Scoring

The results of the model are shown in Figure 11, they are not a competitive result for the Credit Scoring Challenge. Also the training of the model was not easy, since the accuracy based backpropagation is difficult to measure the performance of each epoch, and is weak against the dataset class imbalance.

The training of a deep network is difficult, specially when you have a class imbalanced dataset such as this. The next model of my list should perform better in this kind of problems, then I discarded the neural network model, and followed with the the Gradient Boosting classifier training.

| Your most recent submission | | | | |
|--|-----------|-----------|----------------|----------|
| Name | Submitted | Wait time | Execution time | Score |
| submission_last.csv | just now | 2 seconds | 1 seconds | 0.837229 |
| Complete | | | | |
| Jump to your position on the leaderboard | | | | |

Figure 11: Multi Layer Neural Network Benchmark

For the Gradient Boosting, I've started submitting a fitted model with the default parameters from the Scikit-Learn, and got the following results shown in the Figure 12.

| Your most recent submission | | | | |
|--|-----------|-----------|----------------|----------|
| Name | Submitted | Wait time | Execution time | Score |
| submission_gb.csv | just now | 5 seconds | 1 seconds | 0.860077 |
| Complete | | | | |
| Jump to your position on the leaderboard | | | | |

Figure 12: Default Gradient Boosting Results

I've followed some of the recommendation hiper-parameters from the competitions forum, mainly the ones from [16], for a XGB:

- min child weight: 10.0
- objective: binary:logistic
- max depth: 5
- max delta step: 1.8
- colsample bytree: 0.4
- subsample: 0.8
- eta: 0.025
- gamma: 0.65
- num boostround or num estimators : 391

Based on them I tried a few of my own, for a Gradient Boosting, and after three days of hiper-parameter tuning, the best classifier achieved as shown in Figure 13, achieved a satisfactory result, but still not so competitive, and bellow the sample submission results.

| Your most recent submission | | | | |
|--|-----------|-----------|----------------|----------|
| Name | Submitted | Wait time | Execution time | Score |
| fs_submission_gb.csv | just now | 0 seconds | 1 seconds | 0.861453 |
| Complete | | | | |
| Jump to your position on the leaderboard ▾ | | | | |

Figure 13: Optimized Gradient Boosting Results

The hiper parameters used were:

- loss: deviance
- learning rate: 0.1
- num estimators : 391
- max depth: 4
- criterion: friedman mse
- min samples split: 1.2
- min samples leaf: 1
- min weight fraction leaf: 0
- subsample: 1.0
- max features: None / all
- max leaf nodes: None / default
- min impurity decrease: 0

3.5 Justification

The Kaggle Give Me Some Credit challenge have 924 entries in the leaderboards, the sample benchmark provided by the challenge is 0.864248, 534 submissions achieved a score greater than 0.86. My previous benchmark was for a score greater than 0.86, my best submission is 0.816453, which is bellow the sample submission.






The Gradient Boosting model is clearly capable of achieving a amazing result [16], the neural network is still bellow the benchmark with a 0.83 score, even though the deep neural network approaches achieved spectacular results, their training is complicated for this kind of problem. Possibly a more complex neural network model, with a better parameter tuning and training it can be capable of achieving results closer to the Gradient Boosting.

4 Conclusion

4.1 Free Form Visualization

People often tend to neglect their debts, Brazil has one of the most complex and complicated financial lending and evaluation systems of the world. With the exponential growth of data around the world, with social networks, mobile applications, urban mobility application, digital bankings, and credit cards, the problem of the credit scoring starts not only with the data available for the institution, but also with the data available anywhere.

A user behavior on the *Twitter* for instance, can be measured as a variable for a possible financial distress. My goal with this project is start studying the area of credit scoring to further offer solutions based on the open world data, by having a challenging dataset with unstructured of few data, like a Kaggle Challenge, it's possible to understand how generate a better feature set without needing to collect more data, and to evaluate and refine a good classifier.

| | | | | | | |
|-----|------|--|---|----------|----|----|
| 356 | ▼ 62 | AdjustedRSquared |  | 0.864250 | 12 | 6y |
| | |  Sample Entry Benchmark |  | 0.864248 | | |
| 357 | ▼ 60 | Anthony Goldbloom |  | 0.864248 | 2 | 6y |
| 358 | ▼ 60 | bamboochen |  | 0.864248 | 1 | 6y |

(a) Kaggle Sample Benchmark

| | | | | |
|--|-----------|-----------|----------------|----------|
| Your most recent submission | | | | |
| Name | Submitted | Wait time | Execution time | Score |
| fs_submission_gb.csv | just now | 0 seconds | 1 seconds | 0.861453 |
| Complete | | | | |
| Jump to your position on the leaderboard ▼ | | | | |

(b) My best results

Figure 14: My Results VS the Benchmark

My challenge's results in the leaderboards are still worse than the sample from the owners of the competition, meaning that I still have much to learn about the field of machine learning, and credit scoring to be able to achieve good results in this kind of competition, and create market viable solutions to credit scoring. Fortunately there is a great community like Kaggle to help practice in real scenarios.

4.2 Reflection

The current study was about understanding how machine learning works for enterprise credit scoring solutions. The dataset selected was a public Kaggle's

challenge credit scoring dataset, with about 300.000 data instances. The complete study compared two different model approaches, one based on Gradient Boosting, and another on Neural networks.

The results obtained were satisfactory for the study purpose but not competitive enough to have a good position on the competition's leaderboard. The Kaggle challenges are great to improve the users machine learning skills, by a competitive, and collaborative way.

The main difficulties encountered was the data, mainly because the field of research and applications is still new to me, and the training of both algorithms which took a long time to do multiple 10-fold-cross validations with each hiper-parameter combination. For a future research there are good tools, like HyperOpt ⁴, and Hyperas ⁵.

4.3 Improvement

The selection of a good feature set, and model determines the best results. There is also some competitors that used both unsupervised learning, and supervised learning techniques, and had good competitive results.

Also, fitting a good model on a big dataset can be time consuming, it took me three days to fit a regular model on the dataset with 80 features, and 150.000 data instances. From the two models I've evaluated, the best model was the Gradient Boosting Classifier.

To further improve the model one can use a libraly like HyperOpt combined with Gradient Boosting, or Keras, to find the best hiper-parameters for each model. Also for the neural network there a better loss function, that can be computed based on the AUC score, and like [5] a combination of a genetic algorithm to further improve the hiper parameters of the model.

References

- [1] Wikipedia. Credit Score, https://en.wikipedia.org/wiki/Credit_score
- [2] David West, Neural network credit scoring models, Computers & Operations Research, Volume 27, Issues 11–12, 2000, Pages 1131-1152, ISSN 0305-0548, [https://doi.org/10.1016/S0305-0548\(99\)00149-5](https://doi.org/10.1016/S0305-0548(99)00149-5). (<http://www.sciencedirect.com/science/article/pii/S0305054899001495>)
Keywords: Credit scoring; Neural networks; Multilayer perceptron; Radial basis function; Mixture-of-experts
- [3] Kraus, Anne & Küchenhoff, Helmut. (2014). Credit scoring optimization using the area under the curve. Journal of Risk Model Validation. 8. 31-67. 10.21314/JRMV.2014.116.

⁴<https://github.com/hyperopt/hyperopt>

⁵<https://github.com/maxpumperla/hyperas>

- [4] David Martens, Bart Baesens, Tony Van Gestel, Jan Vanthienen, Comprehensible credit scoring models using rule extraction from support vector machines, *European Journal of Operational Research*, Volume 183, Issue 3, 2007, Pages 1466-1476, ISSN 0377-2217, <https://doi.org/10.1016/j.ejor.2006.04.051>. (<http://www.sciencedirect.com/science/article/pii/S0377221706011878>)
Keywords: Credit scoring; Classification; Support vector machine; Rule extraction
- [5] K. Tran, T. Duong and Q. Ho, "Credit scoring model: A combination of genetic programming and deep learning," 2016 Future Technologies Conference (FTC), San Francisco, CA, 2016, pp. 145-149. doi: 10.1109/FTC.2016.7821603 keywords: financial data processing;genetic algorithms;learning (artificial intelligence);Australian customer credit data sets;German customer credit data sets;IF-THEN rules;credit risk;credit scoring model;customer lending;deep learning network;financial institutes;genetic programming;machine learning based model;Data models;Genetic programming;Logistics;Machine learning;Sociology;Statistics;Training;Credit scoring;deep learning;genetic programming;machine learning;neural network, URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7821603&isnumber=7821581>
- [6] How to handle Imbalanced Classification Problems in machine learning, March 2017, <https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/>
- [7] Loris Nanni, Alessandra Lumini, An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring, *Expert Systems with Applications*, Volume 36, Issue 2, Part 2, 2009, Pages 3028-3033, ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2008.01.018>. (<http://www.sciencedirect.com/science/article/pii/S0957417408000249>)
Keywords: Bankruptcy prediction; Credit scoring; Ensemble of classifiers
- [8] Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6), 627–635. <https://doi.org/10.1057/palgrave.jors.2601545>
- [9] Gang Wang, Jinxing Hao, Jian Ma, Hongbing Jiang, A comparative assessment of ensemble learning for credit scoring, *Expert Systems with Applications*, Volume 38, Issue 1, 2011, Pages 223-230, ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2010.06.048>. (<http://www.sciencedirect.com/science/article/pii/S095741741000552X>)
Keywords: Credit scoring; Ensemble learning; Bagging; Boosting; Stacking
- [10] Louzada Francisco, Ara Anderson, Fernandes B. Guilherme. Classification methods applied to credit scoring: Systematic review and overall comparison. *Surveys in Operations Research and Management Science*.

- [11] Overview and Applications of Artificial Neural Networks XenonStack
<https://medium.com/@xenonstack/overview-of-artificial-neural-networks-and-its-applications-2525c1addff7>
- [12] Kaggle Competition, Give me Some Credit
<https://www.kaggle.com/c/GiveMeSomeCredit>
- [13] Alex Rogozhnikov, Gradient Boosting explained
http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html
- [14] Sean Owen, Machine Learning: What is an intuitive explanation of AUC? *<https://www.quora.com/Machine-Learning-What-is-an-intuitive-explanation-of-AUC>*
- [15] jmalicki (username), 3rd in the Give Me Some Credit Competition source code: *<https://kaggle2.blob.core.windows.net/forum-message-attachments/1583/model.R>*
- [16] GaryMulder (username), 0.870112 on Private Leaderboard, Kaggle Discussion, *<https://www.kaggle.com/c/GiveMeSomeCredit/discussion/31514>*.