Capstone Project

Machine Learning Engineer Nanodegree

Hyungsuk Kang

September 28th, 2017

Definition

Project Overview

Banks play a crucial role in market economies. They decide who can get finance and on what terms and can make or break investment decisions. For markets and society to function, individuals and companies need access to credit. To prove the credit, FICO[1] score was introduced in 1989 by FICO, originally Fair, Isaac and Company, which is a is a data analytics company based in San Jose, California focused on credit scoring services[2]. Nowadays, Credit Karma is produces free service to look up one's creditworthiness based on their scoring model called 'VantageScore'[3].

However, FICO score does not actually prove that one has no credit. It creates a score by looking at your file from the three major credit reporting bureaus – TransUnion, Equifax and Experian. What's worse, 'VantageScore' model shows rather boasted score and waits for their customers responding to their tailored ads after taking customers' personal information and financial condition[4]. In conclusion, there is no suitable financial model which can determine a person will have financial difficulty in upcoming years and therefore cannot get the loan.

The good service has to provide accurate score for customers to support their financial decision, predicting their future finance. Knowing future is impossible, but it can be approximated with the help of machine learning generated from past cases. In this project, I will compare the performance between machine learning and deep learning to predict the probability which the borrower will face serious financial problem in 2 years, and the optimal model will help borrowers to make the best financial decision. The application uses "Give Me Some Credit" dataset to determine the condition.

[1] *FICO(Wikipedia)*

[2] *Credit Score(Wikipedia)*

[3] *"Are Credit Karma Scores Real and Accurate?" Article by Tim Parker*

[4] *"Why Credit Karma Is Free and How It Makes Money" Article by Basia Hellwig*

Problem Statement

The goal is to create the credit scoring model that will predict the borrower will face a financial difficulty or not. The tasks are involved are the following:

1. Download and preprocess the "Give Me Some Credit" data.
2. Train a classifier that can determine if the borrower is in financial crisis.

Metrics

The model will be evaluated using both F1 score and area under the receiver operating characteristic curve(AUROC). Here is the explanation for each of them.

F1 score is the harmonic mean of precision and recall. Precision is the ratio of true positives to all positive predictions, and it is used to evaluate the model's suitability. Recall is the ratio of true positives to all positive examples, and it is used to evaluate the precision of the model.

$$F_1 Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Where:

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

and

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

AUROC curve is the trade-off for between the true positive rate and the false positive rate as the classifier's threshold is varied. The metric is only used for binary classification. The advantage of AUROC curve to F1 score is that it is not sensitive to imbalanced classes.

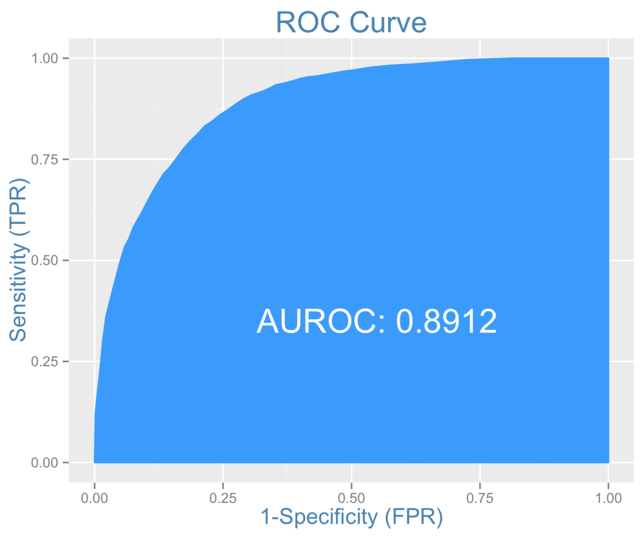


Figure 1. Illustration of ROC curve. Area under the curve is calculated for evaluation.

Analysis

Data Exploration

The "Give Me Some Credit" dataset[5] has the data of 250,000 borrowers with 11 features and 1 label.

Features are following:

- **“RevolvingUtilizationOfUnsecuredLines”**
  Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits
- **“Age”**
  Age of borrower in years
- **“DebtRatio”**
  Monthly debt payments, alimony, living costs divided by monthly gross income
- **“MonthlyIncome”**
  Monthly income
- **“NumberOfOpenCreditLinesAndLoans”**
  Number of open loans (installment like car loan or mortgage) and lines of credit(e.g. credit cards)
- **“NumberOfTimes90DaysLate”**
  Number of times borrower has been 90 days or more past due
- **“NumberRealEstateLoansOrLines”**
  Number of mortgage and real estate loans including home equity lines of credit
- **“NumberOfTime60-89DaysPastDueNotWorse”**
  Number of times borrower has been 60-89 days past due but no worse in the last 2 years
- **“NumberOfDependents”**
  Number of dependents in family excluding themselves (spouse, children etc)

Following label is:

- **“SeriousDlqin2yrs”**
  Person experienced 90 days past due delinquency or worse

There were outliers in each feature. They were detected with percentile based method, which picks the elements in the percentile less than 5% and bigger than 95%.

[5] “Give Me Some Credit” dataset from Kaggle:

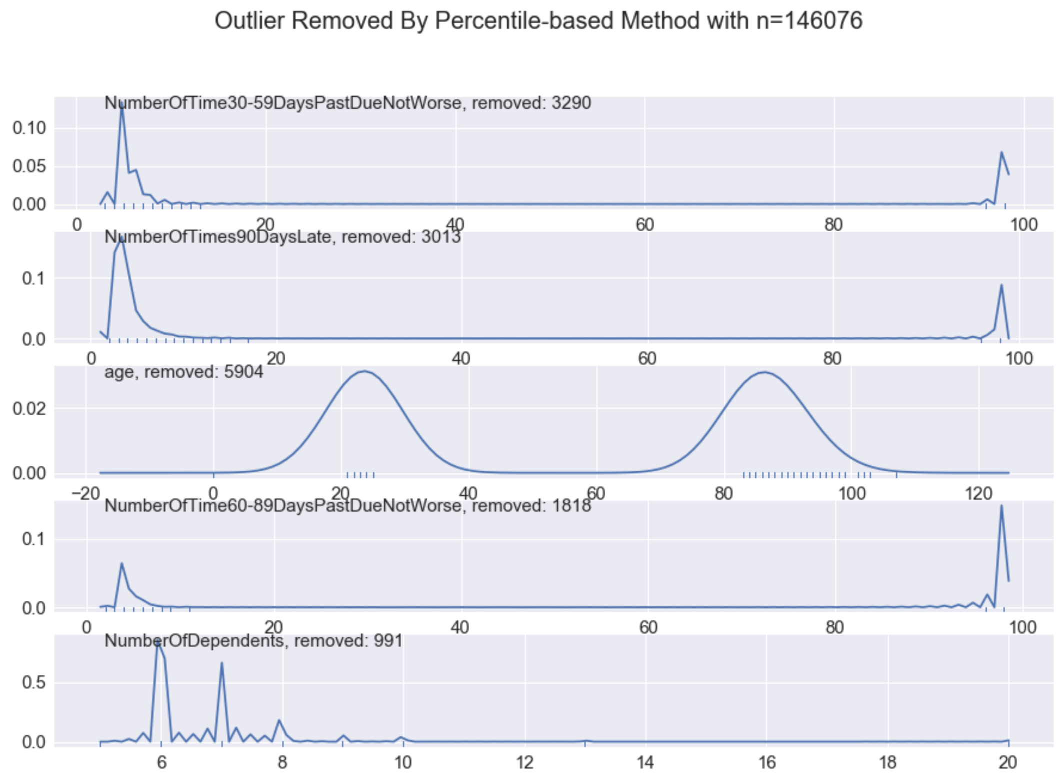


Figure 2. Percentile-based outliers in each feature. Outliers are removed to keep consistency of the model

This plot below shows the distribution of outcome variable in order to get a sense of how balanced or imbalanced is the outcome variable.

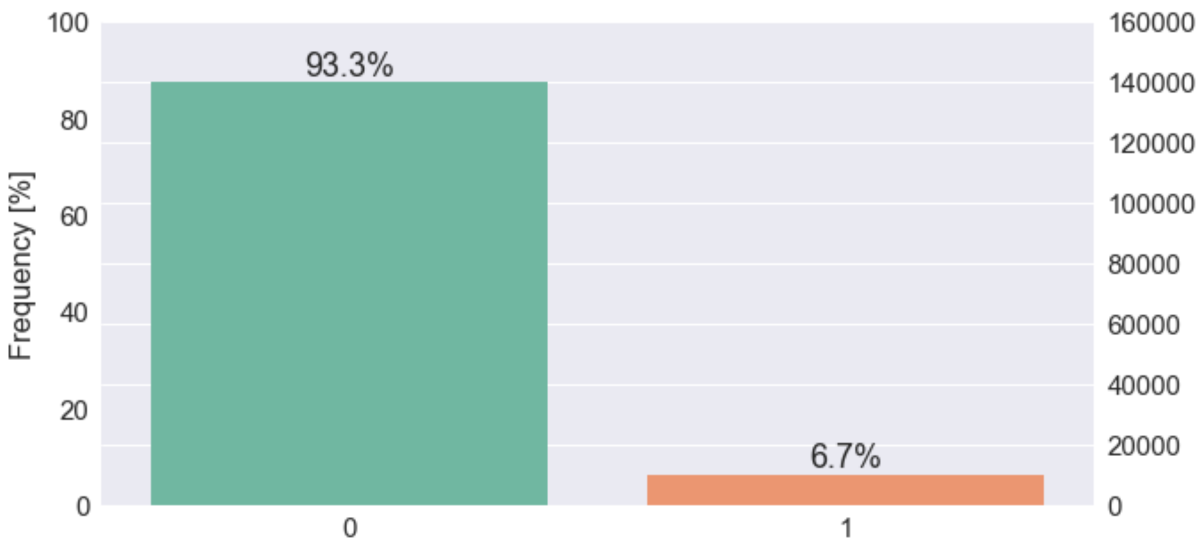


Figure 3. A plot showing the distribution of the outcome variable **"SeriousDlqIn2yrs"**

It can be seen that:

- the output variable is imbalanced to have high bias on 0(not having delinquency or worse).
- Balancing class weight will be required to train each model thoroughly.

Exploratory Visualization

Figure 3 plot shows the heat map of correlation between each feature and the outcome variable. This information is helpful to see the relevance between each feature and the result.

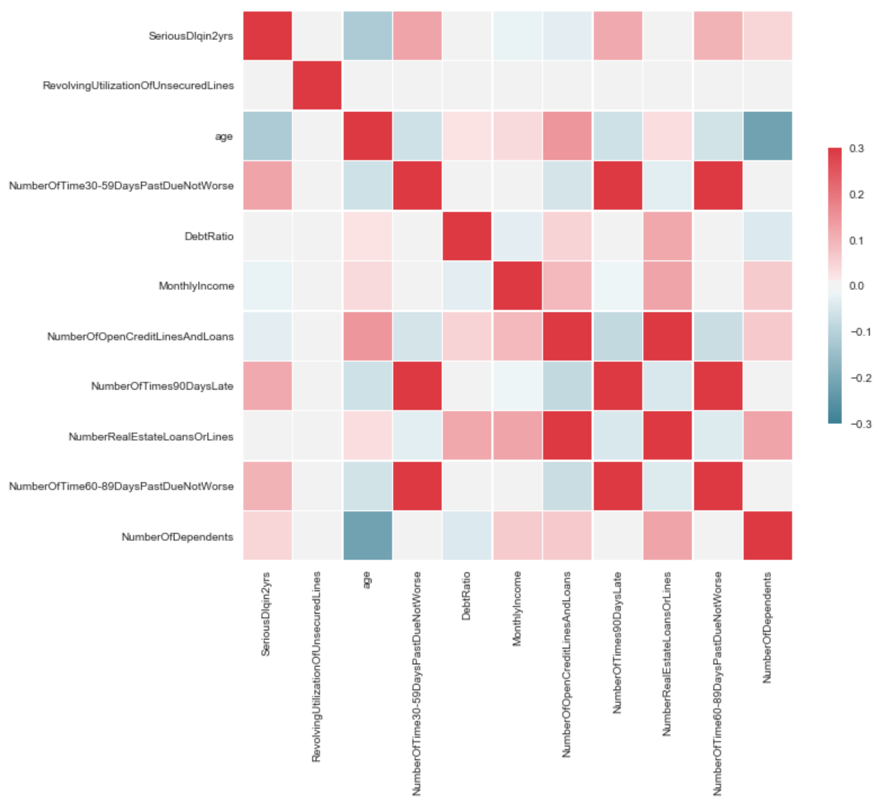


Figure 4. Correlation heat map for each feature including the output variable

The top 5 features which correlates with the outcome variable are following:

1.	NumberOfTime30-59DaysPastDueNotWorse	0.12558
2.	NumberOfTimes90DaysLate	0.11717
3.	age	-0.11538
4.	NumberOfTime60-89DaysPastDueNotWorse	0.10226
5.	NumberOfDependents	0.04604

The model will only use top 5 features to prevent the curse of dimensionality.

Algorithms and Techniques

Machine learning or deep learning algorithms can be implemented to the classifier. Classifiers using machine learning are supporting vector machines(SVM), decistion tree, logistic regression. Classifiers using deep learning is multi-layer perceptron(MLP).

Description for each classifier[6] is following:

SVM

- Supporting vector machine separates data points with decision boundaries with supporting vectors(lines) in multi-dimension data
- Its strength is usually the accuracy tend to be higher than other classifiers
- Its weakness is really long training time

GradientBoostingClassifier

- Based on decision tree algorithm
- Uses greedy algorithm to distinguish elements until the elements are separated without overlapping
- Loss function is calculated with impurity of each category
- Gradient descent is calculated to minimize loss

Logistric Regression

- Logistic regression is the appropriate regression analysis to conduct when the dependent variable is binary
- Returns probability as an output using logistic regression equation $\frac{1}{1+ e^{-(ax+b)}}$ where a, b equals weight and bias respectively
- Weight and Bias is updated from the error between predictions and real values

Multi-layer Perceptron

- Multi-layer perceptron is multi-layer of neurons inspired by the human brain
- Weight is updated with back propagation derived from the error between prediction and real values
- Neural network shows remarkable result to image classification or stock analysis
- Its drawback is that people cannot understand how it really works

Benchmark

To create initial benchmark   for the classifier, I used random classifier. This is same as 'shooting an arrow to the target blindfolded'. However, it provides arbitrary but yet unbiased results.

Methodology

Data Preprocessing

The preprocessing step is done in the Jupyter notebook (titled "Data Analysis and Preprocessing")

The steps are following:

1. Separate top 5 features which is relevant to the outcome variable
2. Drop any NaN data points
3. Detect outlier in each feature based on percentile (x>=95% && x<=5%) and replace them with median value
4. Split data randomly into training and testing dataset

Implementation

The implementation process can be split into two sections:

2. Evaluation stage

During the first stage, the classifier was trained on the preprocessed training data. This was done in a Jupyter notebook (titled “Train and Evaluate the model”), and is further divided into 3 steps:

1. Load dataset
   - Download cs-test.csv, cs-training.csv from https://www.kaggle.com/c/GiveMeSomeCredit/data

2. Test benchmark model
   - Random classifier was selected as the benchmark model.
3. Create a training and predicting pipeline
   - For the train/predict pipeline, I used previous method from supervised learning project(CharityML).
4. Set training parameters for each model
   - Train parameters were chosen for each model:
     *All classifiers’ random state was set to 42 for consistency between models.
     - GradientBoostingClassifier
       Max_depth and subsample was configured from the Kaggle discussion[7]
     - SVM
       Gamma was adjusted to have thinner decision boundary.
       Class weight had to be balanced due to imbalanced output variable distribution.
     - Logistic Regression
       Class weight had to be balanced due to imbalanced output variable distribution.
       Learning rate was set adaptive because smaller learning rate is required to fix minor error as the training continues.
       Max_iter was set to 200 to have better result than default case(100).
     - MLPClassifier
       Class weight had to be balanced due to imbalanced output variable distribution.
       Learning rate was set adaptive because smaller learning rate is required to fix minor error as the training continues.

[7] 0.870112 on the leaderboard:

Max_iter was set to 500 for better result than default case(100).

Shuffle was set true to have unbiased training session.

5. Train/validate the model

   Training and Validation

6. Compare the results with metrics

   Metrics for comparison were:

   - F1-score

   - Area under ROC curve(AUROC)

Refinement

Here are the results of the models from **step 6:**

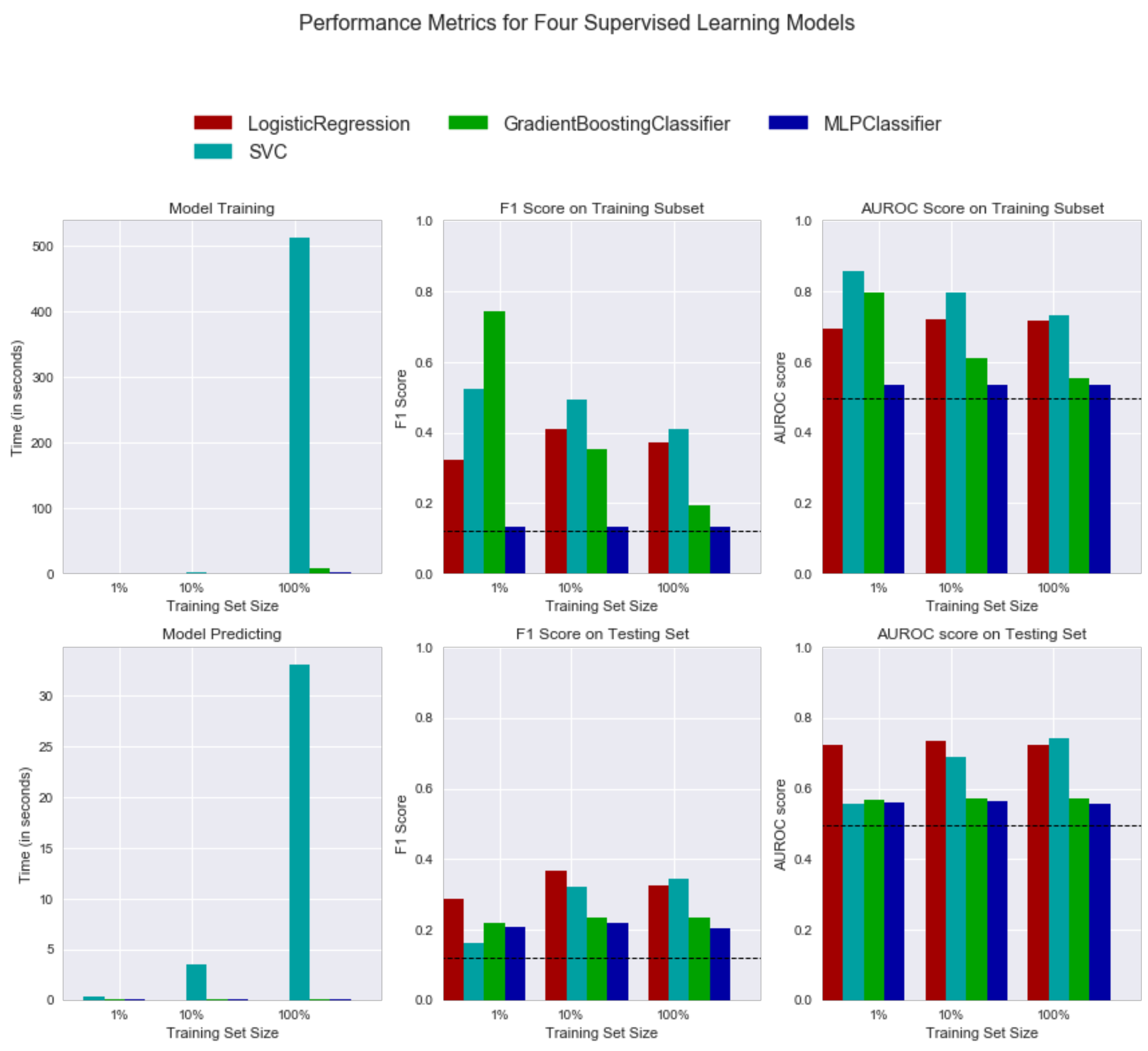


Figure 5. Comparison of machine learning models on the dataset

According to results from 4 models, logistic regression model was not the most accurate model, but training time was drastically shorter than supporting vector machines. As a result, logistic regression model was chosen to tune as the optimal model.

7. Fine tune the optimal model based on the result

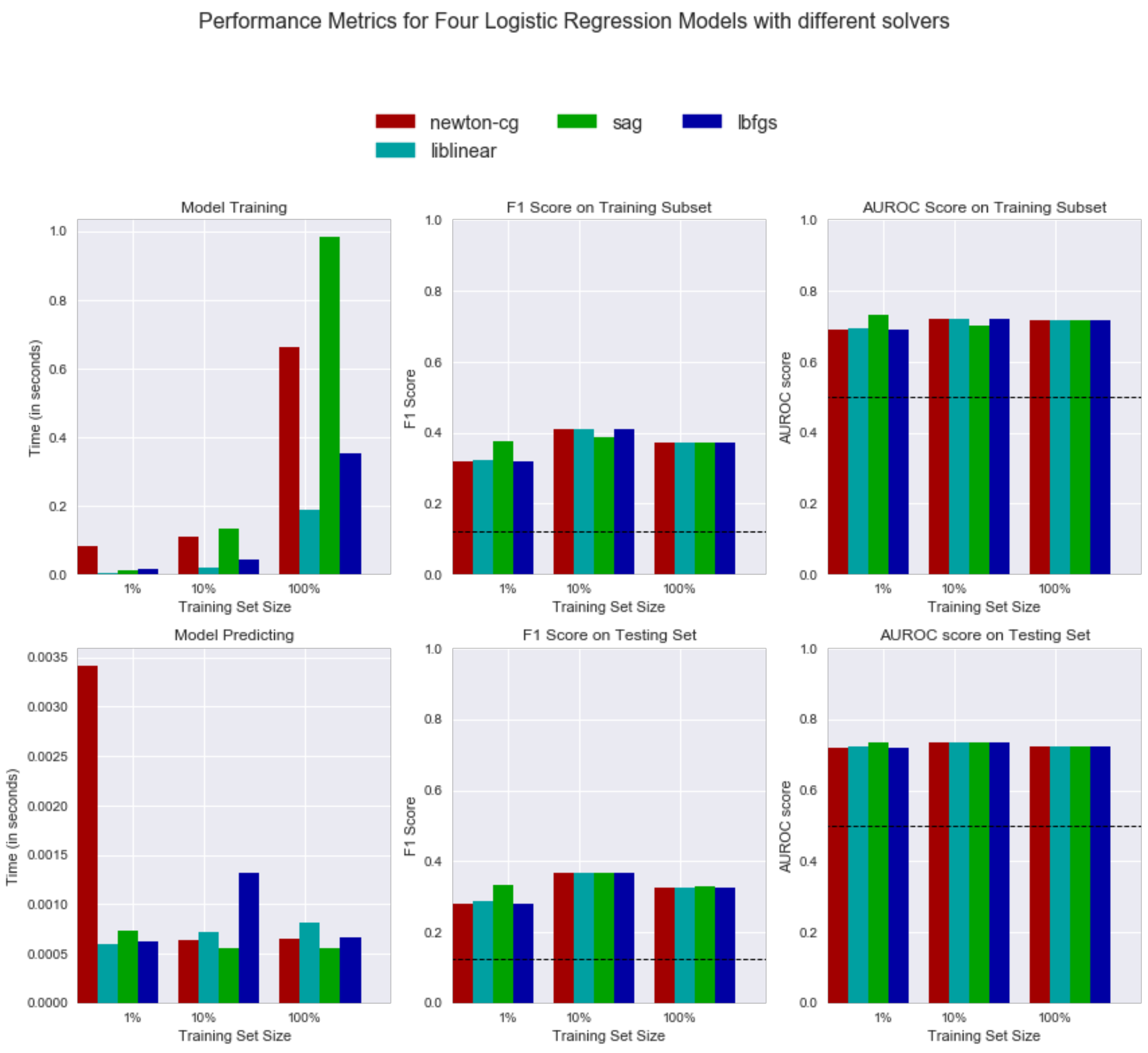


Figure 6. Tuning logistic regression model with maximum number of iteration

The model was tuned with 4 learning method:

- newton-cg
- sag
- lbfgs
- liblinear

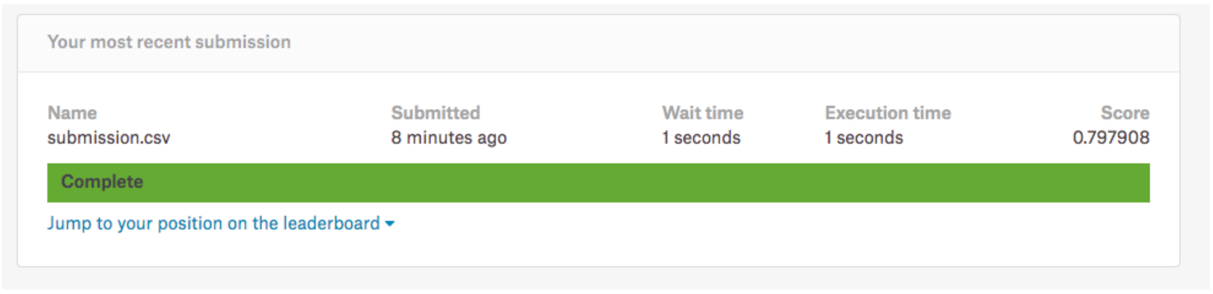
The work of evaluating performance of the model can be found in the notebook (“Fine tuning the optimal model”).

Conclusion

- ❖ Logistic regression was chosen to classify because of its fastest training/testing time and descent F1/AUROC score
- ❖ Class weight is balanced for the imbalanced output variable distribution
- ❖ Its C constant is 1.0
- ❖ Its solver uses liblinear method
- ❖ Its random state is 42

Kaggle Result

Generating result was done in the notebook ("Predicting the financial hardship in the future"), and the AUROC score was about 79.8. This is better than the benchmark score(0.4976).



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission.csv	8 minutes ago	1 seconds	1 seconds	0.797908

Complete

Jump to your position on the leaderboard

Reflection

The process used for this project can be summarized with following steps:

1. Find dataset which is relevant to the problem
2. Preprocess the data (Outlier detection and replacement, drop data points with NaN value)
3. A benchmark model is created
4. Train and validate candidates for optimal classifier using the preprocessed data
5. Tune the optimal model
6. Test the model with test dataset

Improvement

The final model performed better than in the Kaggle discussion[8] with other logistic regression model, but most of competitors used Gradient Boosting algorithm. XGBoost is preferred to scikit-learn. In fact, there are bunch of competition winners who used it[9]. This could

[8] QDA Ensemble Model with .78 score - Code: https://www.kaggle.com/c/GiveMeSomeCredit/discussion/32154

[9] Machine learning challenge winning solutions: https://github.com/dmlc/xgboost/tree/master/demo#machine-

significantly improve the model’s performance with the right hyper parameters. Replacing NaN values with median value of each category with outlier can also improve the classifier with more data without harming the distribution of it.