Porto Seguro's Safe Driver Prediction

Salman Sigari
Haoyue Yu
Boren Lu

BIA 678
Professor David Belanger
Fall 2017

Background

Insurance is one of the most lucrative industries in the world. Insurance firms spend lots of money to have an accurate prediction about their customers to increase their profit. Technology continues to be a huge change driver for consumers, insurers, repairers and the auto industry in general. Whether it is changing how these players communicate, their expectations, repair costs or the vehicles themselves, its impact is wide-ranging, according to the 2017 Crash Course report.

Porto Seguro is the third largest insurance company in Brazil. The company offers services mostly in car insurance. Recently, inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad drivers.

Motivation

In this project, we are challenged to build a model to predict whether a driver will initiate an auto insurance claim in the next year. We have used machine learning methods for the past 20 years to explore more powerful methods. A more accurate prediction will allow the firm to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers. A well-trained model not only will provide an accurate prediction probability of the next year policyholder claim but also will decrease the cost of insurance and also can distinguish between good drivers and the bad drivers. So, the company would sell its insurance products more wisely.

Data Description

The dataset comes from Porto Seguro Insurance Company. It includes both products and customers information of 595,212 observations associated with 58 features with missing values and it has four types of data: binary, ordinal, categorical and continuous. We will explain how to deal with missing value and achieve normalization in the next part. The final dataset consists of 136,398 observations associated with 220 features while we are keeping binary, ordinal and continuous variables.

Data Preparation

Imputing Missing Values

Features with missing value	Feature Type	Missing Percentage
Variable ps_ind_02_cat	Categorical	0.04%
Variable ps_ind_04_cat	Categorical	0.01%
Variable ps_ind_05_cat	Categorical	0.98%
Variable ps_reg_03	Continuous	18.11%
Variable ps_car_01_cat	Categorical	0.02%
Variable ps_car_03_cat	Categorical	69.09%
Variable ps_car_05_cat	Categorical	44.78%
Variable ps_car_07_cat	Categorical	1.93%
Variable ps_car_09_cat	Categorical	0.10%
Variable ps_car_14	Continuous	7.16%

As the table shown above, there are 10 features that contain missing values. Three of them have extremely high missing percentage (missing numbers / total numbers) from 18.11% to 69.09%. Because of the such high missing percentage, one logical solution is simply remove those three features from the dataset.

After removing the features with high missing percentages, there are 7 features remain missing values. Six of them are categorical features and one of them is continuous features. The solution for imputing categorical features is replacing missing values with the mode of each feature itself. And the solution for imputing the only continuous feature is replacing missing values with the mean of the feature itself.

After imputing the missing values, there are 58 variables, not include target variable, in the dataset.

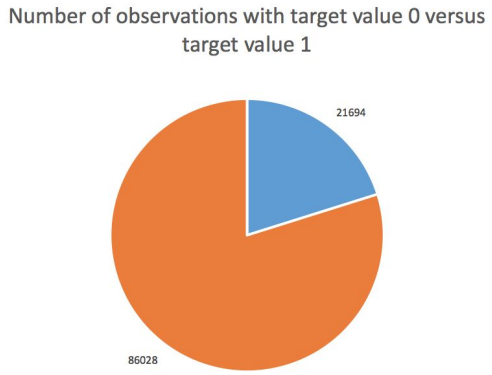
Adjusting Imbalanced Data

The original dataset is terribly imbalanced. 21,694 observations have target value 1 and 573,518 observations have target value 0. The Pie chart below provides an intuitional view of the imbalanced data.

The reason of this imbalanced ratio is that target value 1 means reporting claim and target value 0 means not reporting claim, in the real life, only a small part of all drivers will likely to report claims. Using an imbalanced dataset as a training dataset for machine learning will provide a model with high accuracy but useless in real life. For example, the model would simply predict target value 0 for all the data points in the test dataset and achieving a very high

accuracy. It is obvious that a model that would only predict observations into one class is not useful for business.

For solving this imbalanced dataset problem, 85% of observations with target value 0 are removed from the dataset. The remaining observations include all the observations with target value 0 and 15% of the observations with target value 0 from the original dataset. The new ratio is shown in the below pie chart. The processed dataset is less likely to cause a useless prediction model.



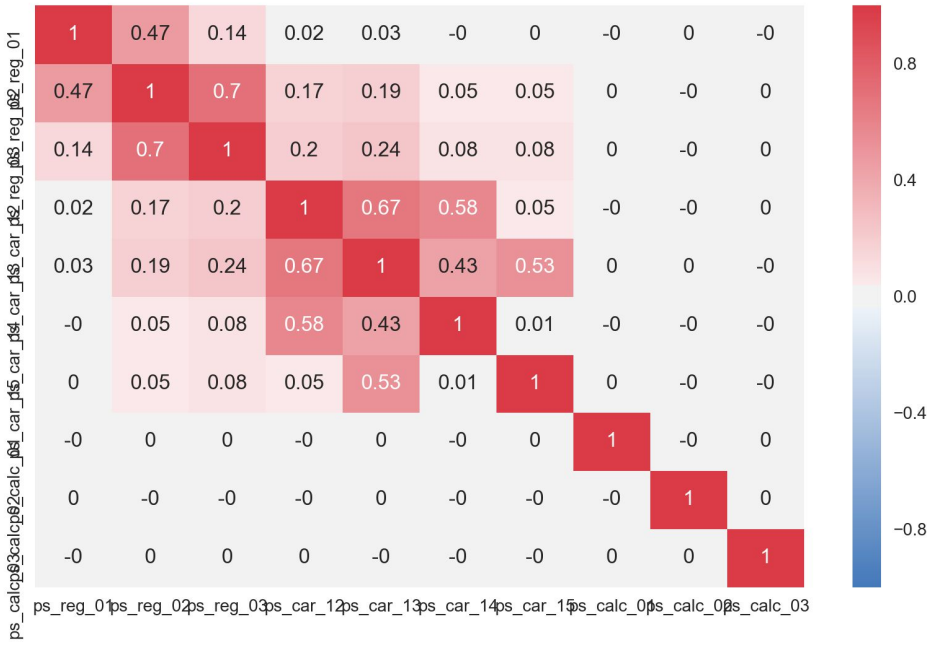
Converting Categorical Variables

There are many categorical features in this dataset. The value in those categorical features represents levels for certain factors. So they should not be treated as the way to continuous variables. All the categorical variables are converted into dummy variables. In this case, if a categorical variable has three levels, then this variable will be converted into three dummy variables which are all binary variables. After converting the categorical variables, this dataset could be feed to models like linear regression as same as other dataset that only contain continuous variables.

But there is a problem for this process. The categorical features are converted into too many features that increased the number of variables dramatically. This would increase the computation time for the machine learning process.

Reducing Data Dimensions

The most effective way to solve the problem brought by creating too many features is to reduce the data dimension.



The above table is a correlation matrix of all the continuous variables in this dataset. If two variables are highly correlated to each other, remove one of those two variable would be a good solution for reducing data dimensions.

Data Overview After Preparation

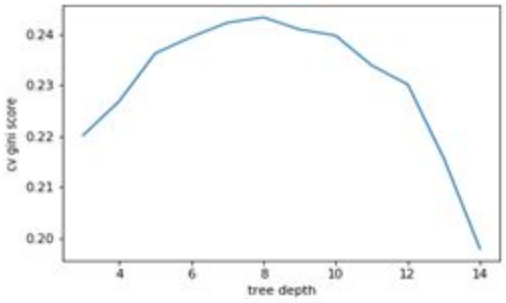
After the data preparation, the dataset is balanced, the number of observations is much less than the original number and the number of features is raised from 58 to 220. Now this dataset is ready for following machine learning steps.

Analysis

We choose Spark as our tool for modeling and analyzing. Spark is one of the most efficient big data processing tools to solve large set of data and achieve real time machine learning. Spark has an integrated framework for streaming engine and machine learning engine. The main library that we used was Mllib, which includes several machine learning algorithms for classifications and measurement.

This project we use supervised learning method, which generates prediction by training the model using our existing training dataset and predict for test dataset. Firstly, we apply cross-validation methodology for training dataset, using random state equals to 42 to split train and validation dataset to test algorithms. The first group of classifiers that we choose are logistic regression, Naïve Bayes and decision tree. Fitting all the training and validation dataset into each model, we calculate TP, FP, accuracy score and AUC, and plot ROC curves for each algorithm.

For the second part of analysis we use ensemble method – random forest for further explore. First of all, by using Geni Score – split nodes based on variables to get the gini impurity criterion for the two descendent nodes, then add up the decreases for each variable to find importance in all trees, we optimize the depth of tree of random forest algorithm. As the picture below indicates, when tree depth equals to 8, random forest performs best on this dataset.



The final step of analyzing is comparison. First of all, we want to compare the performance of our first group classifiers, with considerations of scales. Secondly, we want to analyze whether ensemble method outperform simple classifiers. Lastly, we want to explore scalability performance in parameters and algorithms.

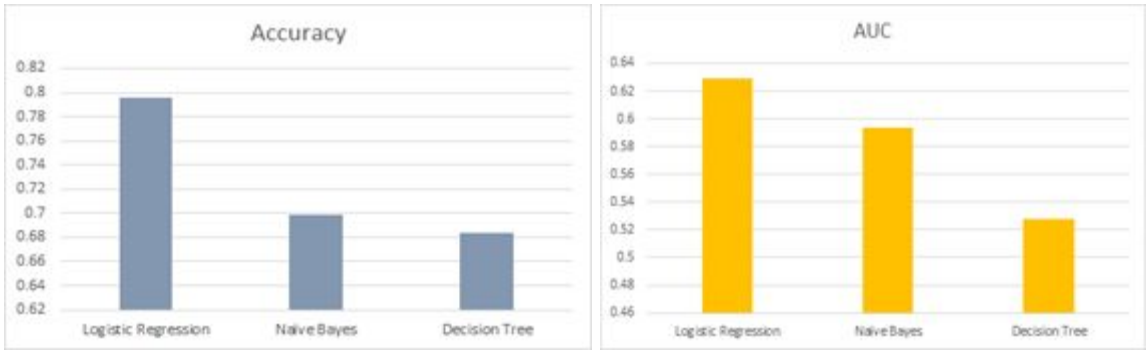
Results

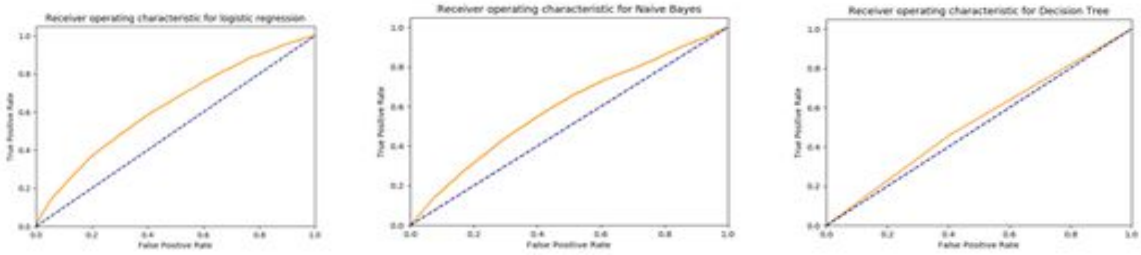
After running three basic algorithms, we get the results that when full dataset is applied, the accuracy rate for logistic regression is around 0.795278. For Naïve Bayes, accuracy rate is around 0.69895. Lastly, decision tree has an accuracy rate around 0.683. For AUC score, logistic regression scores around 0.629. Naïve Bayes scores around 0.594, and decision tree only has AUC of 0.528. From the result here, we discover that logistic regression has the best performance among these three classifier. Since we have a lot of missing values and dummy variable, decision tree does not perform well.

Performance Measurements

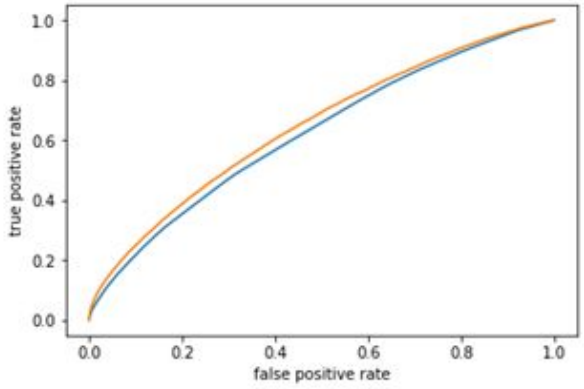
Algorithms

As the results above, we first to use accuracy rate, AUC and ROC curves to compare these three basic methods:





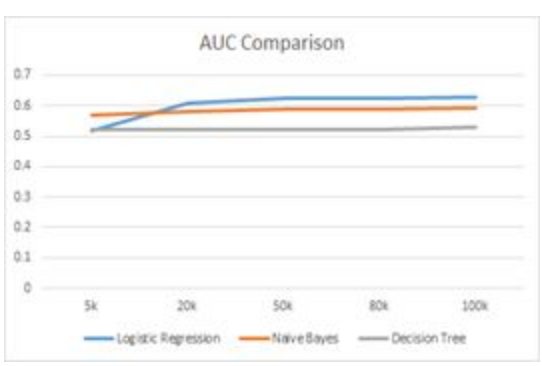
From these graphs, we see that logistic regression has the best performance regarding to accuracy rate, AUC and ROC curve. Then we choose logistic regression to compare with random forest:

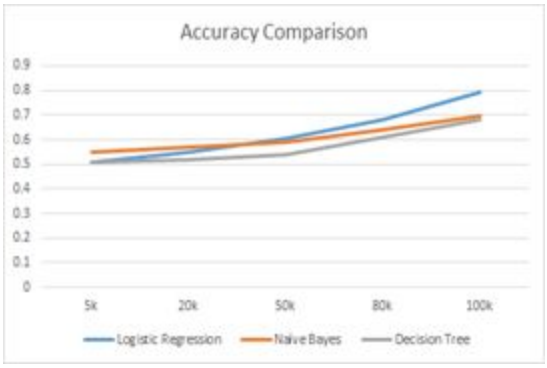


In this graph, orange line indicates Random forest while blue line indicates logistic regression. As we see, random forest outperforms logistic regression regarding to AUC and ROC curve.

Scalability/Parallelism:

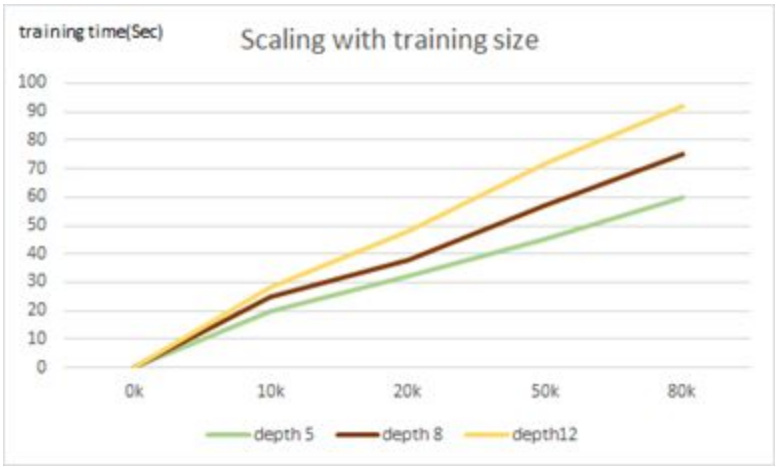
In order to decide the scalability performance, we subset of dataset from 5k to 100k to find out trends. First of all, we analyze the scalability within basic classifiers:



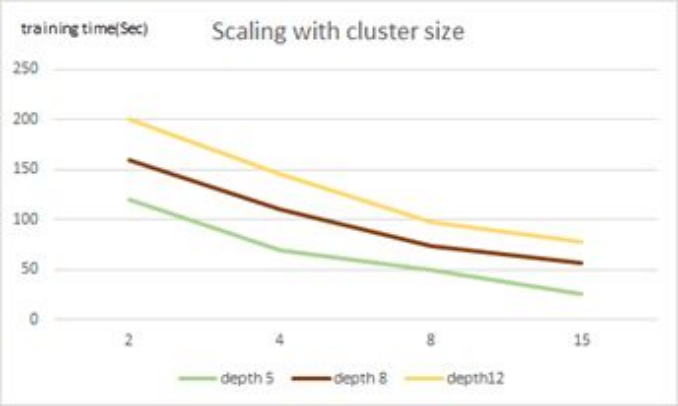


Those two graphs explain that while increasing the size of dataset, we can get better model performance. To be noticed, logistic regression has a greater increase than decision tree and Naïve Bayes. When the dataset is smaller, Naïve Bayes has a more stable and better performance than other algorithms. Logistic regression has a better performance while dealing with larger set of data.

As for random forest, we use different number of tree depth, sizes of dataset and the number of clusters to measure scalability and parallelism performance. Setting tree depth as five, eight(the most optimal), and twelve, we measure the time of training as we increase the dataset.



As we see, the more data that the model takes, the more time it takes to train the model. Besides, if we increase the number of tree, the training time will increase at the same time and this phenomenon is clearer when the dataset is large. To test parallelism performance in random forest, we choose the number of clusters and training time as measurements.



Increasing the number of clusters can decrease of training times of the model, while the more depth of trees we use the more time it can take. To sum up, scalability and parallelism are two significant factors in big data analysis. Using larger set of data and more clusters not only can increase performance of algorithms but also can build models more efficiently.

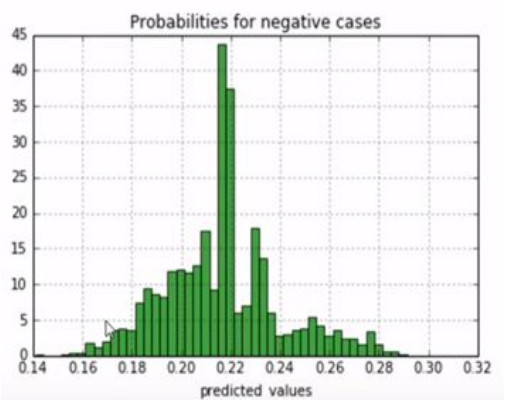
Unbalanced Strategy

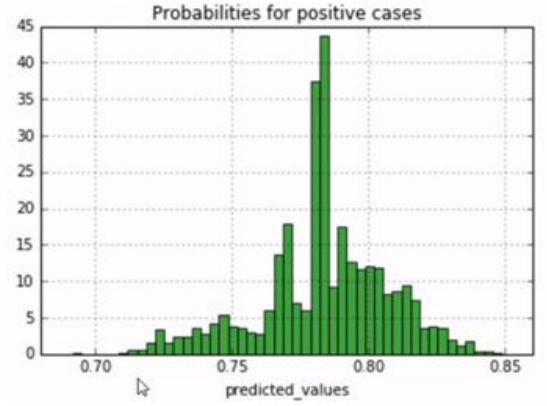
As we mentioned before, the dataset is unbalanced because our classes (positive and negative cases) are represented equally. Unequal distribution of dependent variable causes the performance of existing classifiers to get biased towards majority class. The challenge here is differentiating positives from negatives. We can bring down the ratio between positive and negative cases with two strategies. First, we applied down-sampling manually. We specified the ratio of positive and negative cases by ratio adjustment in pyspark and applied ratio = 4. It means

that for each positive cases we have 4 negative cases and after running the algorithm we got the ROC score of 0.64633.

```
''' Down-sampling: reduce sampling for positive cases'''
from numpy.random import randint
from pyspark.sql.function import udf
from pyspark.sql.types import IntegerType

RATIO_ADJUST = 4.0 ## ratio of negative to positive in sample
```






```
The ROC score is (@numTrees=200):  0.6463328674547113
```

For the second strategy, we considered ensembling paradigm for Random Forest. Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. We have considered ensemble of six times down-sampling for this problem and from the predictions, and we calculate the average. The improvement after each run represented in the below figure.

```
from pyspark.mllib.evaluate import BinaryClassificationMetrics as metric

RATIO_ADJUST = 3.0 ## ration of negative to positive in the train_df
TOTAL_MODELS = 6
```

```
Round:  0
The ROC score is (@numTrees=200):  0.6456296366007628
Round:  1
The ROC score is (@numTrees=200):  0.6475210701955153
Round:  2
The ROC score is (@numTrees=200):  0.6488169677072237
Round:  3
The ROC score is (@numTrees=200):  0.6490333812262444
Round:  4
The ROC score is (@numTrees=200):  0.6490997896881725
Round:  5
```

Conclusion

Based on AUC, ROC curves and accuracy score, we can conclude that Random Forest has a higher score, which means that it performs better on our insurance prediction dataset. Besides modeling, when dealing with high dimensional datasets, we can use feature selection methods to extract insights from the features, such as using RandomforestClassifier to obtain the importance of the features.

Spark, as a powerful tool for big data, is ideal in this project, with mllib and metrics for performance. It is more efficient than local Python to solve large and complicated dataset. Through the project, we also notice that scalability is an important factor in modeling. Increasing datasets can not only improve performance significantly but also can select better algorithms and parameters. Based on our predictions, it is believed that it would be applicable for the insurance company to classify their customers based on the probability of reporting claims. Furthermore, it is a good starting point to construct a risk preference system for current clients. Finally, obtaining more related data, such as the compensation amount for each claim would add more business values to the model, which could significantly improve the use of the risk preference system and classify customers into more accurate groups based on their risks.