Porto Seguro Safe Driver Prediction

Sai Krishna Karthikeya Dulla[1], Shreyas Rewagad[2], Nilima Sahoo[3]

Abstract
Insurance companies needs to predict the chances of a person applying for claims, based on the individual's profile. The goal of this project is to build a model that predicts the probability that a driver will initiate an auto insurance claim in the next year. In this paper we present various data exploration, wrangling & modelling techniques that would help us determine the probability of the individual to file for an auto insurance claim. The paper demonstrates a detailed comparison within the different modelling approaches. Thus, ending up with the best to determine the solution to our problem.

Keywords
Class Imbalance — Naive Bayes — Logistic Regression — Decision Tree — Random Forest — XGBoost — Gini Index

[1] *saidulla@iu.edu*,[2] *srewagad@umail.iu.edu*,[3] *nsahoo@iu.edu*
Data Science, School of Informatics, Computing & Engineering, Indiana University, Bloomington, IN, USA

Contents

Introduction

Predicting the probability of claims is very important for an insurance company to modify accordingly the insurance plans they provide for individual drivers. Porto Seguro, one of Brazil's largest auto and homeowner insurance companies, completely agrees with the fact that inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones. In this project we are trying to build a model that attempts to accurately predict the probability that a driver will possibly make a claim in the next year.

1. Data

The data is collected from the Kaggle data-sets by Porto Seguro. It contains the details about the claims and a target variable - `target`, that tells whether the individual has applied for a claim or not. The data is mostly clean and well formatted, also we do not have to do much about the integrity of the data as this data is procured from the Kaggle website.

As in case of most insurance data-sets our data-set also consists of missing values. We have explored several methods to deal with the missing values, which are discussed later under *Data pre-processing*. Furthermore, we have chosen a set of algorithms that would help us model the data and procure the probabilities for the positive class, which in our case is $target = 1$, i.e. the customer has filed for an insurance claim. To determine the effectiveness of our models we have resorted to using Gini coefficient as our evaluation measure, which is also being used by this competition to predict scores. Gini score can be calculated as $Gini = 2 * AUC - 1$, and we know that Area under Curve (AUC) is a good evaluation metric for a class imbalanced dataset, which ours is.

1.1 Data Exploration & Pre-processing

1.1.1 Data Description

The data provided by the insurance company is posted in Kaggle. It is already divided into train and test. The data dictionary helps us with the variable information, stating the class of the different variables. We have 4 different types of variables - ind, reg, car & calc. We do not have more information about this, here is further bifurcation of these types as

present in the train & test data -

# Train data-points	595212
# Test data-points	892816
Total Variables	59
Response/Class Variable	target
ind	Total: 18
	Categorical: 3
	Binary: 11
	Continuous: 4
reg	continuous: 3
car	Total: 16
	Categorical: 11
	Binary: 0
	Continuous: 5
calc	Total: 20
	Categorical: 0
	Binary: 6
	Continuous: 14

1.1.2 Missing Data

Exploring the data we get to know about the existence of missing values. Enlisting the distribution of missing values in the respective data-sets -

Train		Test	
ps_ind_02_cat	0.04	ps_ind_02_cat	0.03
ps_ind_04_cat	0.01	ps_ind_04_cat	0.02
ps_ind_05_cat	0.98	ps_ind_05_cat	0.98
ps_reg_03	18.11	ps_reg_03	18.11
ps_car_01_cat	0.02	ps_car_01_cat	0.02
ps_car_02_cat	0.001	ps_car_02_cat	0.001
ps_car_03_cat	69.09	ps_car_03_cat	69.10
ps_car_05_cat	44.78	ps_car_05_cat	44.84
ps_car_07_cat	1.93	ps_car_07_cat	1.94
ps_car_09_cat	0.10	ps_car_09_cat	0.10
ps_car_11	0.001	ps_car_11	0.001
ps_car_12	0.005	ps_car_14	7.15
ps_car_14	7.16		

It appears that both Test & Train data-sets have similar distribution w.r.t. missing values. Also, certain variables exhibit a huge percentage of missing values. We need to get rid of such variables as imputing these may be potentially equivalent to introducing noise in the data. Hence, we remove `ps_reg_03`, `ps_car_03_cat` and `ps_car_05_cat`. For rest of the variables we have imputed the missing values utilizing the below strategy:

case i. Replace the missing values in categorical variables with the mode values of those corresponding columns. For each continuous variable with missing values we identified a continuous variable with which it has high linear correlation and fitted a linear model to estimate the missing values.

case ii. Replace the missing values in categorical variables with mode values of corresponding columns and continuous variables with mean values of corresponding columns. This approach turned out to be better than case(i), because the estimated values from linear fit are giving slightly higher mean squared error.

case iii. The missing values in continuous variables are still replaced with mean values of corresponding columns. But in the case of categorical variables, instead of replacing the missing values we treated the missing value also to be a new category in each variable, which gave a significant boost in the error rate. The reason for this could be that even the commonly missed details of drivers might be an useful information while deciding the claim probability.

case iv. The missing value in categorical variables is considered as a separate category as in case(iii). The missing values in continuous variables have been imputed using *KNN* imputation[1] with $k = 5$, i.e. for every observation to be imputed, it identifies k closest observations and computes the weighted average (weighted based on distance) of these k observations.

All the above mentioned approaches are tested using different models, the results of which can be seen in later stages.

1.2 Exploratory Data Analysis

To do feature engineering and find the features which are more important, we performed few exploratory data analysis techniques. We define **claim rate** to be the proportion of claims grouped to that particular value of a variable in the data, i.e. for example for a specific variable V1, suppose we have 100 records with category as "a" out of which 10 records have target value as 1, which means that the claim rate for category "a" in variable V1 is $10/100 = 0.1$ and percentage claim rate = 10%. We tried to plot individual features with the percentage claim rates to know how much impact does values in a feature have in predicting the claims. We will also see how the distribution of claims looks like.

First of all, we plotted claim rates with respect to the categorical features. It can be found in Figure 1 and Figure 2.

Our finding was, few of the categorical features are skewed for example, ps_ind_02_cat, ps_ind_04_cat, ps_car_01_cat. Visually, it can be seen as around 50% of data is falling under -1 category which are the NA values.

Next we looked into the binary features. The plots can be found in Figure 3 and Figure 4.

In the plots, lightblue bars are the TRUE or "1" values and darkblue are FALSE or "0" values. For Figure 3, we can see significant difference in claims being claimed and not claimed. But, in Figure 4, we do not see much of a difference. The binary values are almost equally distributed for ps_calc_15_bin, ps_calc_16_bin, ps_calc_20_bin, ps_calc_17_bin, ps_calc_18_bin .

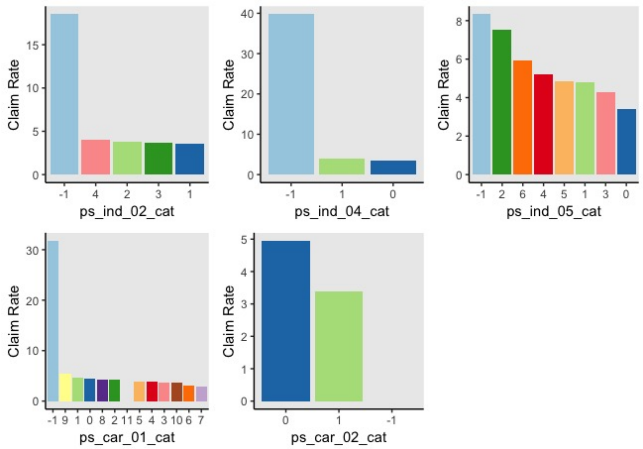

Figure 1. Categorical features w.r.t. Claim Rate

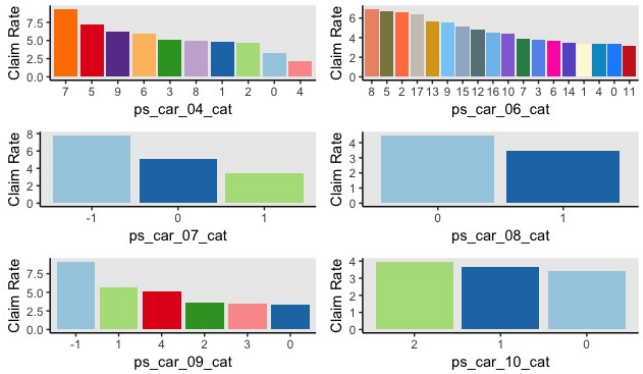

Figure 2. Categorical features w.r.t. Claim Rate

For the ordinal features we plotted combined plot of line plot and scatter plot. This way the variation of that feature can also be seen.

In figure 5, we see an overall claim rate values lie between 3 to 20%. For ps_ind_01, the claim rate increases with increase in the feature values whereas for ps_ind_15, it is related negatively. For ps_ind_03, the claim rate reached 6% when the value is zero and decresed gradually. In Figure 6, we have rest of the ordinal variables plotted. There is a significant variation in between the variables with respect to Claim rate. Especially for ps_car_10, ps_car_11 and ps_car_13, the claim rate decreases at around 20% after being is a constant range of 4%.

Overall after seeing the features and it's impact on target variable, we could see strong differences in claim rates in different groups. The strongest impact on claim rates is done by categorical and integer features. In particular the "ind" and "car" variables. It's variation range is from 2% to 3%. The calc integer values as well have neutral effect on claim rate suggesting that the calc group in general is not of immediate usefulness for our prediction goal.

The binary features has also have some effect on claim rates in the range of 1-2 percentage points, whereas others show no practical impact. Most of the "ind" features have high impact on claims whereas the "calc" binary features are

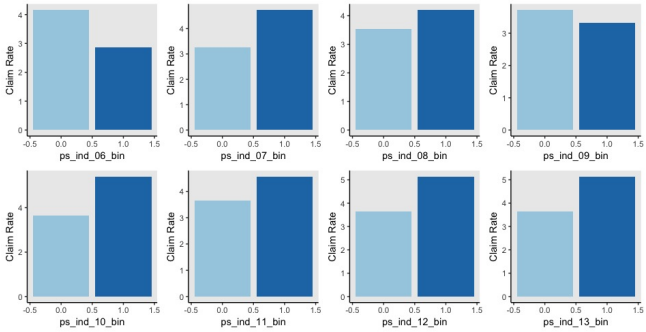

Figure 3. Binary features w.r.t. Claim Rate

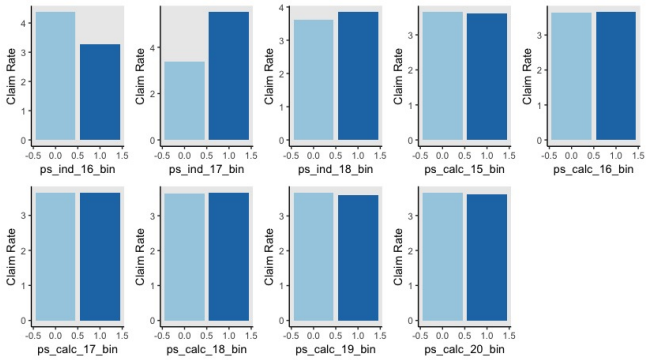

Figure 4. Binary features w.r.t. Claim Rate

neutral.

Next we tried to find correlation between the independent variables. It is plotted in Figure 7. From the plot, it can be observed that, there is not much correlation between the variables in the leftmost column. There is a strong correlation between ps_ind_14 and ps_ind_12_bin i.e. a value of 0.94. ps_ind_18_bin and ps_ind_16_bin are also somewhat correlated. ps_car_13 and ps_car_15 have a correlation value of 0.68 and ps_reg_02 and ps_reg_03 have also the same correlation coefficient as 0.68. The correlation between the "reg" and "car" features shows how continuous variables are related. ps_ind_16_bin and ps_ind_18_bin are negatively correlated with a value of -0.58.

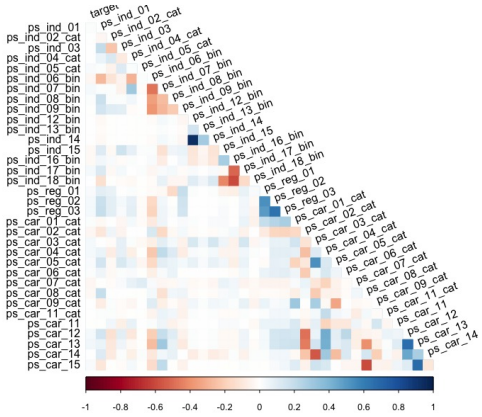

Figure 7. Correlation plot between the independent variables

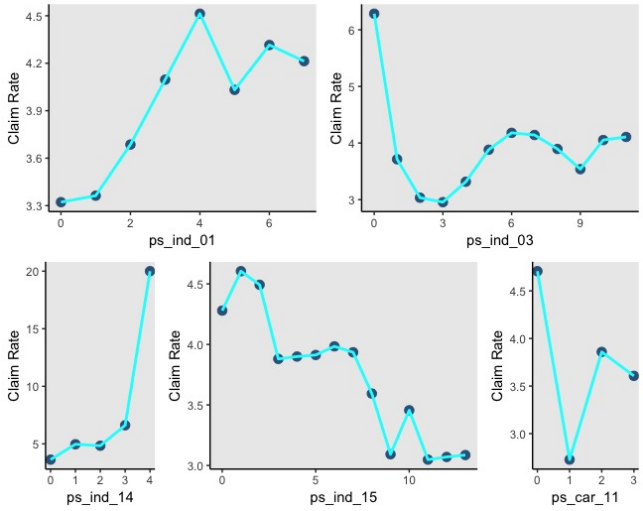

Figure 5. Integer features w.r.t. Claim Rate

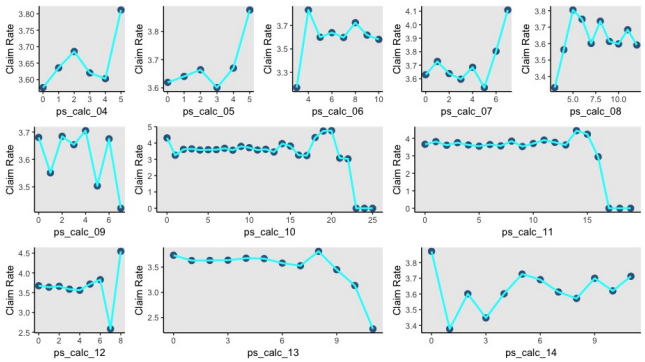

Figure 6. Integer features w.r.t. Claim Rate

2. Algorithm and Methodology

The task here is binary classification, though we need to compute the probabilities of a person initiating a claim, i.e. the probability of observation belonging to class 1. We have started the modelling with a basic algorithm such as Naive Bayes, and later used Logistic Regression, Decision Trees, Random Forests and XGBoost. The performance comparisons of these algorithms on the current dataset is presented in later sections.

2.1 Naive Bayes

Bayes classifier gives a better understanding of the classification problem from a statistical perspective, it tries to maximize the posterior probability of a class for a given object using the prior and class conditional probabilities. The way Bayes classifier is defined, makes it an optimal classification technique from the statistical point of view.

Naive Bayes classifier is a variant of the Bayes classifier, where it assumes independence between variables. This becomes a very good representative of Bayes classifier when the independence assumption holds approximately fair for the data. From the initial investigations on the dataset, we observed that most of the variables are uncorrelated. Though independence criterion is difficult to find in a practical scenario, variable being uncorrelated gives a good reason to proceed with this algorithm.

2.2 Generalized Linear Models

Generalized Linear Models (GLM) are a flexible generalizations of ordinary linear model, i.e. it extends the linear modelling framework to variables that are not normally distributed. It houses various statistical models, such as Linear Regression, Logistic Regression, Poisson Regression etc.

2.2.1 Logistic Regression

In statistics, logistic regression, is a regression model where the target variable is categorical. This is a good model for binary classification. Using the setup with Binomial distribution and link function as *logit* in the GLM models will use the logistic regression for binary classification.

2.3 Tree Based Modelling

The Tree based modelling techniques are amongst the most widely used techniques for classification. These do not required any prior assumption of the data. Though, these produced skewed results when there is a class(target variable) imbalance and also, these methods tends to capture every bit of variation in the data resulting in the over-fitting problem. Often considered while implementing a tree based model, we consider tweaking the below hyper-parameters: Minimum samples for a node split, Minimum samples for a terminal node (leaf), Maximum depth of tree (vertical depth), Maximum number of terminal nodes, Maximum features to consider for split, etc., we are able to successfully evade the over-fitting issue by tweaking there hyper-parameters by performing k-fold cross-validation. Also, pruning the tree is considered to help us get around over-fitting. Lets discuss the various tree based modelling techniques we have experimented with.

2.3.1 Decision Tree

Decision Tree - as discussed above, this is a Non-Parametric method which does not constraint/limit us on the kind of data-type of our attributes. Also, we especially do not need to handle missing data to make this technique work. As in the case of every tree based model, the decision tree also works on the principal of reduction in variance in the data. Meaning to say that the best/most homogeneous split is observed when the reduction in variance in the resulting data-set is the least. We opt for different ways to determine this by: Information Gain, Gini-Index & Chi Square.

2.3.2 Random Forest

This is a type of ensemble learning method, which is a consolidation of a group of weak models to form a powerful model. Random Forest essentially divides the data into sub-samples these are subsets of data in term of #attributes and #columns. Trees are grown on these sub-sampled data-sets. And their aggregation is done to achieve prediction. Random Forest doesn't employee pruning but works well with missing and

maximum of 53 levels in a categorical attribute - which is a standard of most random forest packages in R. Working with the sub-sampled data(bootstrap sampling) helps reduce over-fitting. The technique helps determine the variable importance, thereby utilizing/assigning higher weights to important attributes and penalizing attributes that do not help increase the predictive power of the model.

2.3.3 XGBoost

Over the random forest method xgboost also has a range of add-ins. We are able to perform regularization both L1 and L2, this helps reduce over-fitting. The algorithm also employees a range of objective functions which help in dealing with most types of data. The implementation is in parallel, thus making the entire classifier-learning process fast. XGBoost has pruning built in. It makes splits up to the `max_depth` specified and then start pruning the tree backwards and remove sub-tree that result in no positive gain. The R package has even built in cross-validation to help determine the optimal parameters for model-fitting.

3. Experiments and Results

3.1 Naive Bayes

The initial experiments were started with modeling the data using Naive Bayes classifier. As most of the variables in the data set are uncorrelated, we assume that the independence assumption holds reasonable to use this algorithm. The pre-processed train dataset is divided into training(80%) and test(20%) data subsets and is modeled with the Naive Bayes algorithm with Laplacian smoothing. The obtained model is used to predict the available test data, which resulted in a Gini index score of 0.221.

3.2 Logistic Regression

To improve on the baseline created previously, we opted to use the Generalized Linear Models setup with logistic regression to predict the response variable in terms of probability values.

The dummy variables are created for all the categorical variables using onehot encoding, and the variables resulting in linear combinations are removed as they could cause rank deficiency in the model. This model increased the Gini score to 0.257. Later we tried a k-fold cross validation using Generalized Linear Models taking $k = 5$, and used the parameters with least error on the cross validated test data, but this doesn't seems to have any impact on the final test data. It resulted in same Gini score.

Though we observed a significant improvement using logistic regression when compared to Naive Bayes. We observed that it was very difficult to tune this model. We tried different data processing techniques which only had a slight improvement in the performance. So we were inclined to try a new model.

3.3 Decision Tree

An intuition behind using the tree based approach to modelling this data was that these algorithm would work well despite the kind of distribution that the data. Proceeding with the model we get to observe a gini score of 0.1451361, which is quiet inferior with respect to our earlier considered algorithms. We know for a fact that Decision tree is going to produce skewed results in case of class imbalance. But, performing further experiment we see that even for balanced classes our results fail to improve. One observation that we make from the Smote technique is that, the over sampling of the positive class 1 fails to deliver appropriate data-points. Thus making it harder for decision tree to decide good homogeneous splits.

3.4 Random Forest

Judging by the mechanism of the random forest algorithm, we think that growing a forest would help minimize the misclassification by penalizing the trees that are weak predictors. Though, it appears to be a reasonable thing to do, we are set-back by the limitation of the algorithm. As the implementation is not in parallel, the minimum number of trees required to help learn the appropriate model to the classifier has never achieved do to hardware limitation. Being serialized, it takes awfully long to train the classifier. Restricting us to know the optimal hyper-parameters. Thus, rejecting this approach.

3.5 XGBoost

Modelling using XGBoost[2] algorithm gave a significant improvement the performance. We got a Gini score of 0.267 just from using the default values for hyper-parameters. Unlike logistic regression, tuning the hyper-parameters seems to have done a good job in this case. We attained a Gini score of 0.280 with the tuned parameters.

3.6 Handling Class Imbalance and Model Comparison

This section contains some of the experimental results of different approaches we followed. Main problem in this dataset is the high skewness in class distribution, we tried to down sample the majority class which didn't produce better results, so we tried SMOTE[3] algorithm which balances the data by doing both down sampling and up sampling, though it produced better results than just down sampling it did not produce better results than the original unbalanced data. This can be observed in the Figures 8; for all the models we can see that Gini index on balanced data is lower than the Gini index for unbalanced data.

These plots also display the performance improvement in each model for different data processing steps we have done. The x-axis is each plot contains five cases, four of which *case_1*, *case_2*, *case_3* and *case_4* are different ways of handling NA values that we have discussed in the Missing Data section above. The fifth case is defined as below.

case v. The *calc_bin* variables are removed from data. As we have described in the Exploratory Data Analysis section

the *calc_bin* variables are not impacting the claim rate, i.e. they are not a good predictors. After removing those variables, the missing data is handled using approach case(iii) which is the best among those four approaches.

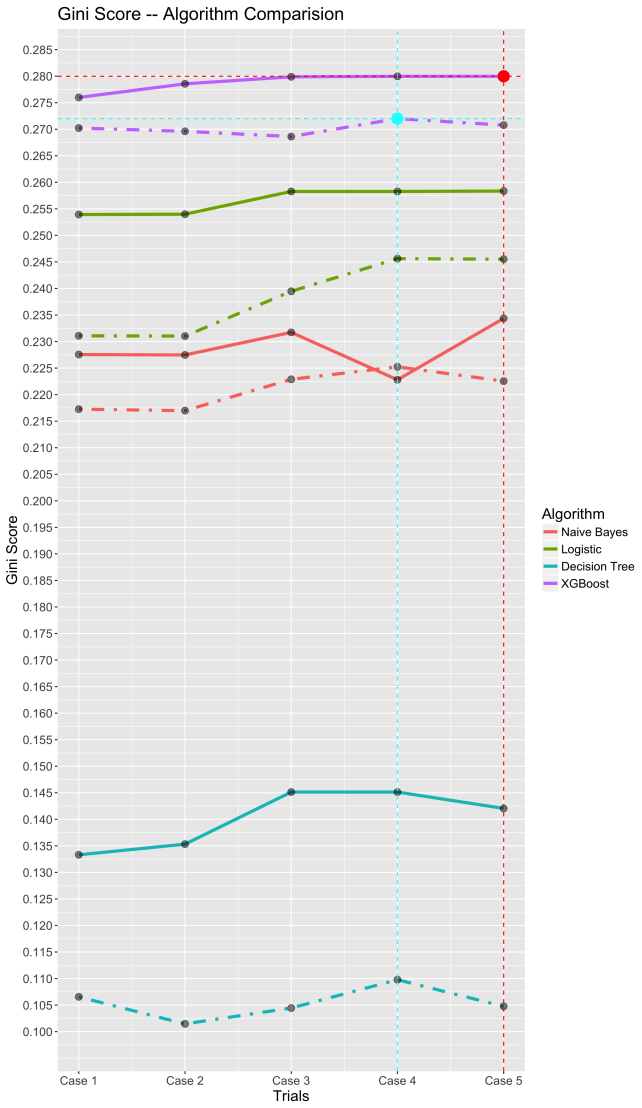

Figure 8. Algorithm Comparision – Gini Score w.r.t. the different data preprocessing approaches for Class Imbalance(Line) & Smote Class Balance(Dotted-Line)

Among all the pre processing approaches for different models *case_5* seems to be doing a good job. Next to it, *case_3* and *case_4* are performing better, which have been using missing values in each categorical variable as a unique category to itself, seems like a good assumption compared to other NA handling techniques. However, there is no considerable difference between the performance of *case_3* and *case_4*, which tells that *KNN* imputing does not have much of an impact.

We can also observe the performance comparison of the three models we used in Figure 9, which has the Gini scores plotted for a 5-fold cross validation for each model. Clearly xgboost outperforms the remaining models by a large margin.

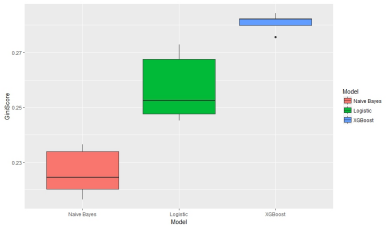

Figure 9. Cross Validation Gini Scores across models

4. Summary and Conclusions

In this project, we extensively investigated the dataset. We experimented with different data cleaning strategies and carried out different techniques to model the data. The major challenge endured in this project was determining features that would help increase the predictive power for our positive class. It is imperative to reduce the false negatives as the cost of a successful claim failed translates to a heavy loss to the insurance company. The XGBoost model appears to be performing well as it helps reduce these cases.

Acknowledgments

In this project we explored many Machine Learning techniques and implemented them as well. We would like to thank our Professor Hasan Kurban for his support and advice all throughout. We would also like to thank our Associate Instructors for their timely technical guidance and feedback. It has been a great learning experience.

References

1. Torgo, L. (2010). Data Mining with R, learning with case studies Chapman and Hall/CRC. URL: http://www.dcc.fc.up.pt/ ltorgo/DataMiningWithR
2. Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. doi:10.1145/2939672.2939785
3. Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W. 2002. SMOTE: Synthetic minority oversampling technique. Journal of Artificial Intelligence Research. 16, 321-357.
4. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Introduction to data mining. 1st, 2005.
5. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning, volume 112. Springer, 2013.
6. H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.