ID5059 - Practical 2 Individual Report

180024570

19/04/2019

Contents

1 Introduction

The report summarises the data mining project to analyse the car insurance data from a large auto and homeowner insurance company in Brazil, Porto Seguro, and build a model that predicts the probability that a driver will file an auto insurance claim in the next year.

1.1 Datasets

The train data has 595212 observations, the test data has 892816 observations, and each row corresponds to a policy holder. Both dataset contains 59 variables.

The *target* column in the train dataset signifies that a claim was field (value of 1) or not (0), which is the predicting target for the train dateset, and the overall claim rate is about 3.64%. Other columns in train and test datasets are in the same structure:

- There are four groups of features, tagged as *ind*, *reg*, *car*, and *calc*, respectively indicating information about individuals, regions, cars and calculated features.
- The postfixes of feature names indicate the data types, *bin* for binary, *cat* for categorical.
- Those without the postfix are either ordinal or continuous features.
- Missing values are labelled as value of -1.

1.2 Objective and Evaluation

The project objective is to build a model prodicting the probability that a policy hold will file an claim in the next year and identify the key features that impact the probability. The *target* outcomes for the test data are required to be values within the range of 0 to 1, instead of boolean values as in the train data. Model performance is evaluated by normalised Gini coeffiecient, accuracy, AUC, while the predict results will be evaluated by normalised Gini coeffiecient ultimately. Particularly, the public Gini score on Kaggle is based on 30% of the test data, while the private score is based on 70%. The final model selection will be based on the private score as the official competetion rule stated.

2 Methodology

2.1 Exploratory Data Analysis and Processing

In the training dataset, 846458 out of 595212×59 data items are missing values. Several processing methods for missing data are implemented to test the water, including imputation with mean values or mode values, regarding missing values as a specific level of that category, or simply eliminating the record.

As shown in figure 1, the features *ps_car_03_cat* and *ps_car_05_cat* has large proportions of missing values, hence, removing records that contain missing values results in massive loss of training data, which is obviously not a good choice. The provisional models based on other two methods suggest, that including *NA* vulues as a particular level for categorical features tends to fit better models. This is consistent with figure 2, which indicates that *NA* as a category somehow is impacting the claim probability to some extent. For instance, records with some features missing have much higher claim rates than those records with complete data for that feature. Therefore, in the following model fitting stage, the missing data are treated as a specific level for that feature.

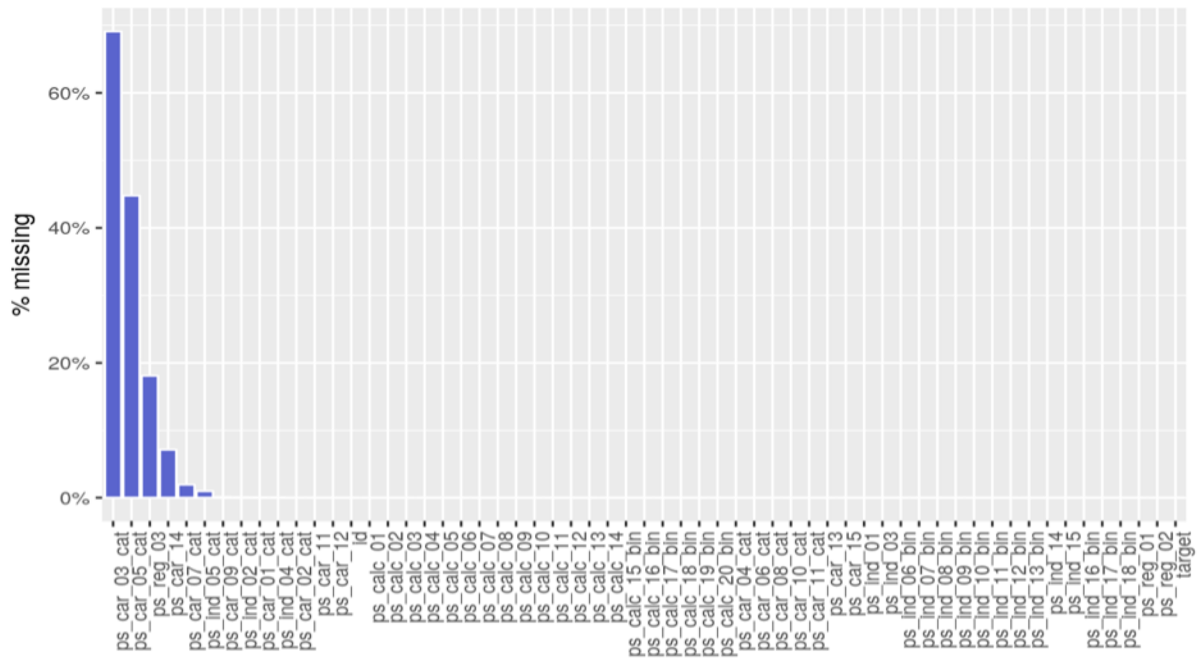


Figure 1: Missing Values by Feature

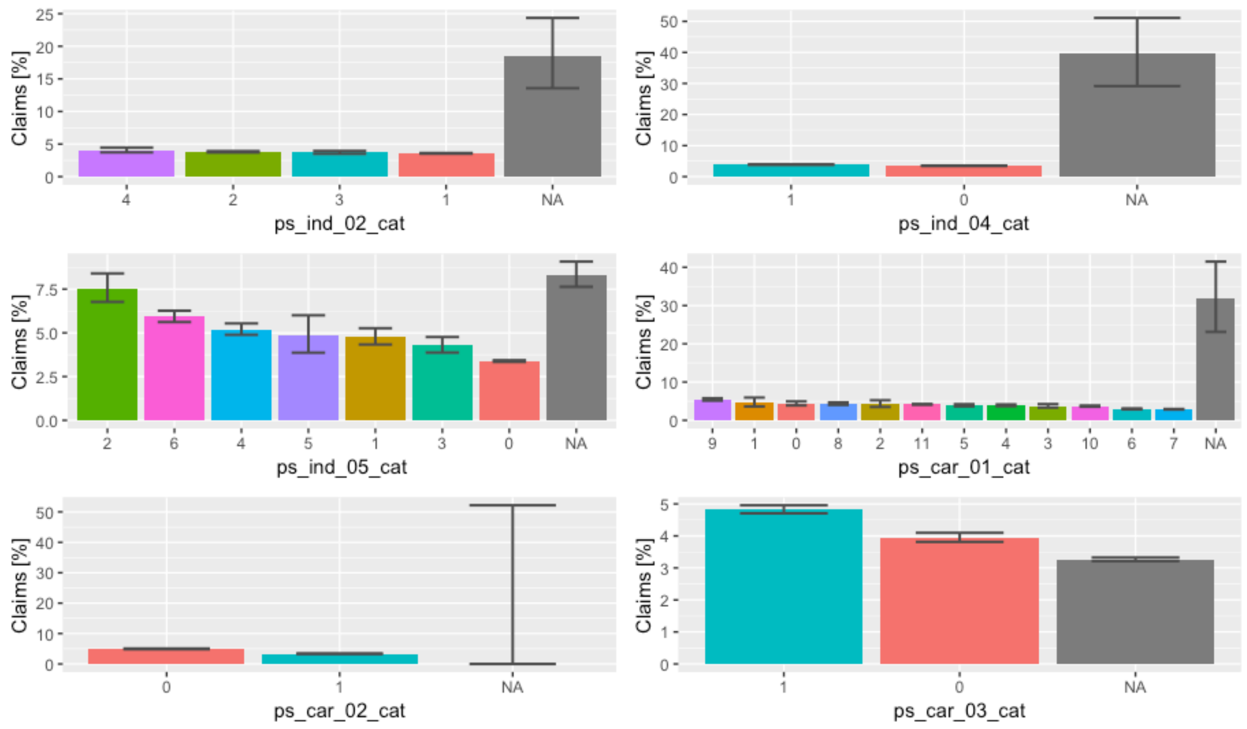


Figure 2: Impact of Missing Values

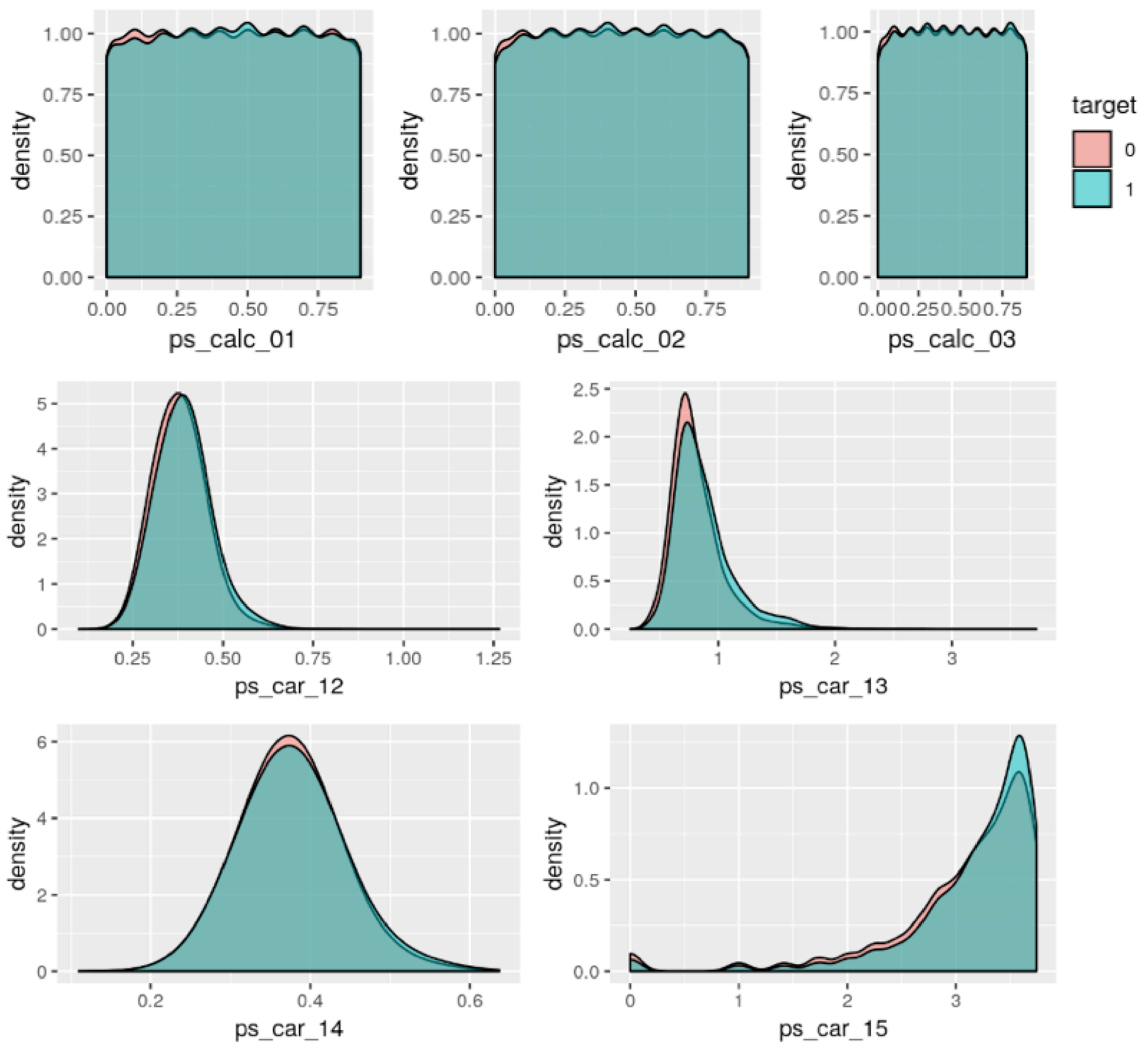


Figure 3: Trivial Influence of Calculated Features, Kaggle Community (2018)

Figure 3 sugguests that the calculated features with suffix *calc* barely have any influences on the claim rates and hence will be eliminated before the model fitting stage.

The categorical and binary features with suffix *cat* and *bin*, are encoded into dummy variables, and the data frame is then converted into a sparse matrix for model fitting. The test dataset is processed in the identical way to the training dataset.

2.2 Model Fitting and Selection

Our group has implemented four machine learning techniques to fit models, including random forests, logistic regression, neural nets and gradient boosting. Each of us undertakes one method and aims to attain the optimal model of the corresponding method via hyper-parameter tuning and grid search. The rest of the report will be focusing on my part, which is gradient boosting. All works are implemented using R Core Team (2018) and on 2.6 GHz Intel Core i7 computer with 16 GB 2133 MHz memory.

Extreme Gradient Boosting algorithm by Chen *et al.* (2019), a.k.a XGBoost, is chosen for the implementation of gradient boosting method, because of its efficiency, flexibility and capability of parallelized computing. A dozen models are fitted across the whole process. However, for length and simplicity reasons, four models are selected to be presented in this section, and each of them represents one phase of the model fitting process.

The training dataset is *randomly* split into two subsets for exploratory fitting in the early stage. One subset contains 80% of the total training data, and the other contains 20%. The two subsets will be referred as *subset80* and *subset20*.

In the early stage,

- phase one, fitting with default parameters. Representative: Model 1, trained with *subset20* , *subset80* as validation data;
- phase two, fitting with adjusted parameters. Representative: Model 2, trained with *subset20* , *subset80* as validation data.

In the later stage,

- phase three, grid search. Representative: Model 3, fitted with full training data and evaluated by normalised Gini coefficients based on 5-fold cross validation;
- phase four, hyperparameter fine-tuning. Representative: Model 4, fitted with full training data and evaluated by normalised Gini coefficients based on Kaggle leader board.

Table 1: Evaluation of the Four Representative models

	train Gini	test Gini	Public Gini Score	Private Gini Score	test Accuracy
model1	0.95507	0.14083	-	-	0.96313
model2	0.49764	0.25553	-	-	0.66880
model3	0.32181	0.27843	0.27956	0.28395	-
model4	0.31987	0.27738	0.28243	0.28641	-

The evaluation information about the four representative models are illustrated in table 1 above.

Please note that for efficiency reason, models are trained with a small proportion of the whole training data in the early stage and validated with the rest unseen part. No ultimate predictions of test data is generated for submission, therefore the public and private scores are not available for model 1 and 2. In the later stage, models are fitted with the complete training dataset and generates predictions of the test dataset for submission. However, the actual target data of test dataset is not available, therefore the test accuracy cannot be calculated for model 3 and 4, and the train Gini and test Gini are based on 5-fold cross validation.

Table 2: Model 1 Confusion Matrix based on the validation dataset (subset80)

	0	1
0	458588	17322
1	234	26

Model 1 is trained with the default parameters of the algorithm and results in extreme overfitting, and this is reflected by the large difference between its train Gini and test Gini. The confusion matrix of model 1 shown in table 2, indicates that it almost predicts all results to be 0, which leads to a particularly high accuracy even based on the validation dataset. Obviously, this high accuracy is meaningless because of the inherent unbalance of the training dataset, in which 96.36% are negative. The model is extremely bad at predicting high risk policy holder, since only 26 out of (26 + 17322) are successfully identified.

Table 3: Model 2 Confusion Matrix based on the validation dataset (subset80)

	0	1
0	309728	8614
1	149094	8734

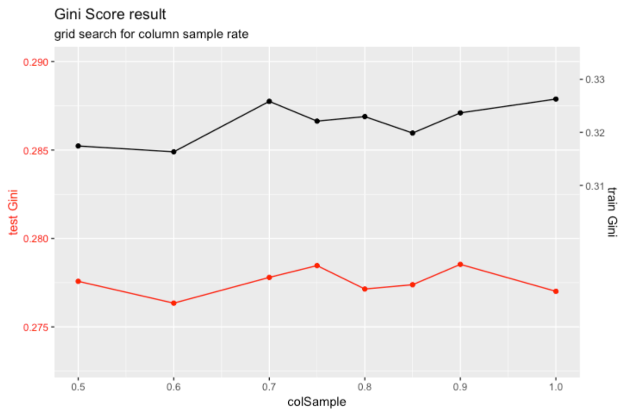
In phase two, models are fitted with an addtional parameter set as the radio of negative to posive in training dataset, and this can control the balance of positive and negative weights. As shown in table 3, the confusion matrix of model 2 indicates that approximate half of the high risk policy holders are correctly identified, which is much better than model 1 in phase one. The difference between train Gini and test Gini has decreased due to some intuitive adjustments to the parameters controling the learning rate, maximum number of trees, subsampling of columns, etc. However, the overall accuracy also has a distinct decrease because a vast amount of true negatives are wrongly predicted to be positive.

In the later stage, the whole training dataset is used to fit candidate models for finer parameter tuning, and the evaluations rely on submissions to Kaggle for returned Gini scores, as well as 5-fold stratified cross validations.

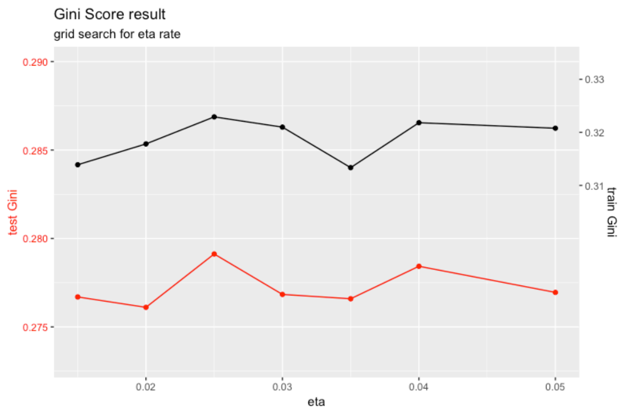
In phase three, a couple of grid searches have been implemented and the results are depicted in table 4 and figure 4, where train and test Gini scores are based on 5-fold stratified cross validations. In addition, the evaluation function is custimized as normalized Gini, and the stopping rule is set to be stop if the test Gini score did not improve for 30 rounds.

Table 4: Grid Search Multiple Parameters

trainGini	testGini	eta	maxDepth	colSample	min_child_wt
0.3019270	0.2736922	0.010	3	0.8	5
0.3223452	0.2769021	0.025	3	0.8	5
0.3233942	0.2745375	0.010	4	0.8	5
0.3438172	0.2777177	0.025	4	0.8	5
0.3646612	0.2772371	0.010	5	0.8	5
0.3670230	0.2760990	0.025	5	0.8	5
0.3019711	0.2742799	0.010	3	1.0	5
0.3251433	0.2763914	0.025	3	1.0	5
0.3258536	0.2750885	0.010	4	1.0	5
0.3414396	0.2765634	0.025	4	1.0	5
0.3669564	0.2777857	0.010	5	1.0	5
0.3705014	0.2749472	0.025	5	1.0	5
0.3018225	0.2736750	0.010	3	0.8	15
0.3249463	0.2776639	0.025	3	0.8	15
0.3325422	0.2772209	0.010	4	0.8	15
0.3402828	0.2763466	0.025	4	0.8	15
0.3595766	0.2766231	0.010	5	0.8	15
0.3689881	0.2743106	0.025	5	0.8	15
0.3019097	0.2729169	0.010	3	1.0	15
0.3289177	0.2774156	0.025	3	1.0	15
0.3327171	0.2769325	0.010	4	1.0	15
0.3483596	0.2769012	0.025	4	1.0	15
0.3569824	0.2762935	0.010	5	1.0	15
0.3583241	0.2750904	0.025	5	1.0	15



(a) search for column subsampling rate



(b) search for learning rate

Figure 4: Grid Search for Single Parameter

Grid search for multiple parameters at one time takes tens of hours to complete, due to the large amount of data and combinations. However, it is hard to figure out a pattern or identify the optimal parameter combination from the result in table 4, even the local optimal could be randomness. Single parameter search results can be easily illustrated with plots, but the trend is still not clear, and repeating the searches with different random seeds did not return stronger results either. Nevertheless, the overfitting has been further mitigated as shown in table 1.

Since the grid search in phase three did not return prominent trends, in phase four model parameters are manually fine-tuned within relatively optimal ranges. Every fitted model generates preditions for test dataset and submit to Kaggle for ultimate evaluation. The best model at the end is model 4 with private score of 0.28641.

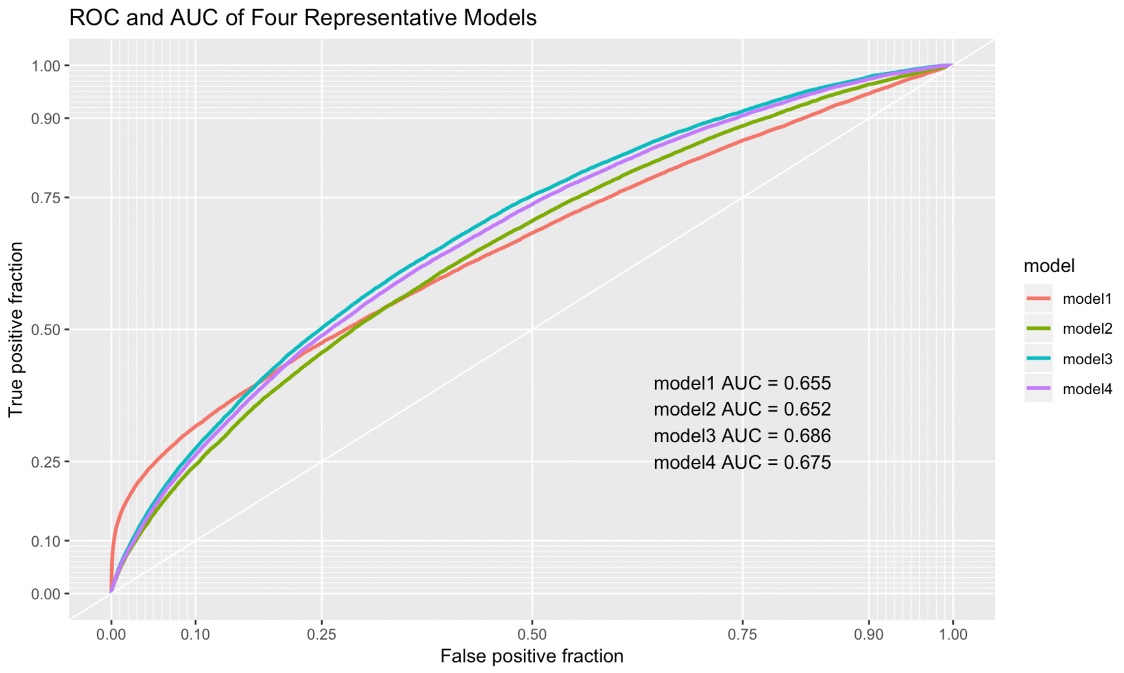


Figure 5: ROC Curves and AUC Scores of the Four Representative Models

ROC curves based on the test dataset cannot be computed due to the lack of actual results. Instead, the ROC curves and AUC scores based on the training data are illustrated in figure 5. Model 1 is clearly biased due to the ignorance of the unbalanced target. Model 3 has perceptibly better curve and higher AUC score than model 4, the leader board scores, however, suggest that model 3 is slightly overfitted than model 4. Therefore, model 4 has been chosen to be the final representative of gradient boosting method, as well as the model of the group according to figure 6 and table 5.

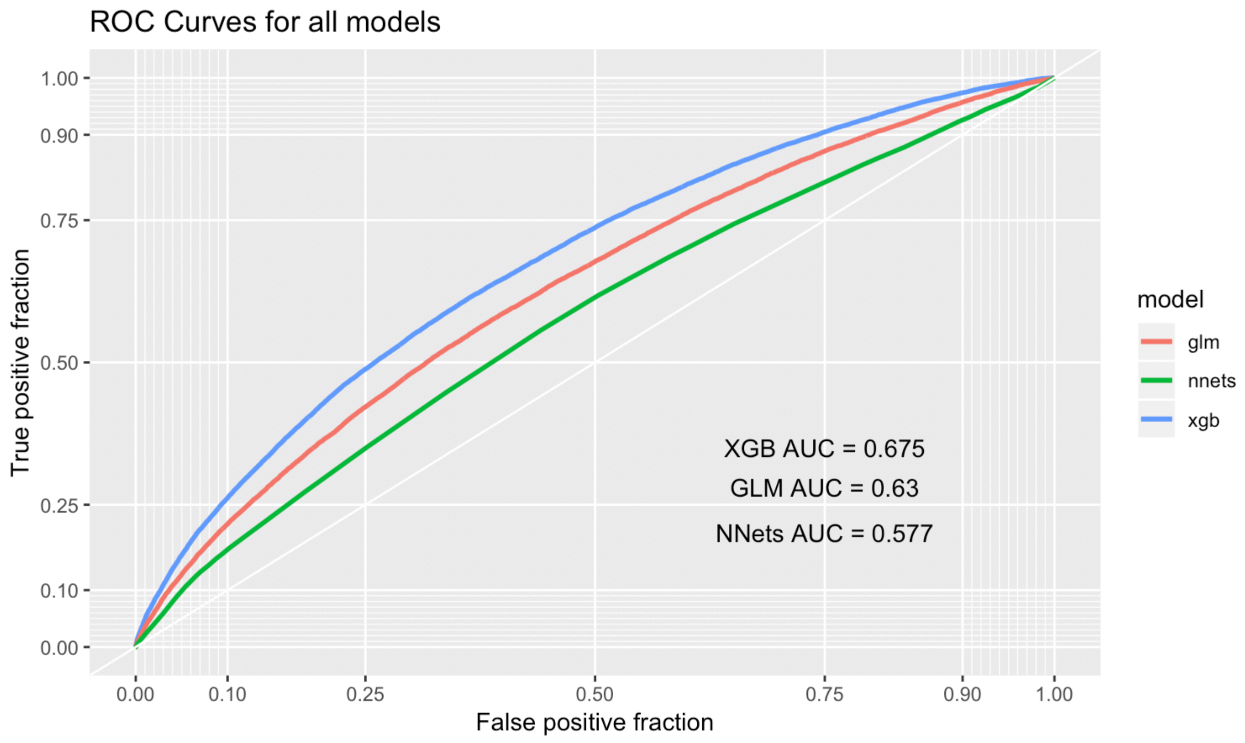


Figure 6: ROC Curves and AUC Scores of the Group Models

Table 5: Normalized Gini Scores based on Test Data

	Public Gini	Private Gini
Xgb	0.28243	0.28641
GLM	0.23079	0.23761
NNets	0.13324	0.14371

3 Reflection

Data processing took a significant part of the workflow. Because of the complexity of real-life data, feature engineering can sometimes be even more influential than techniques or algorithms. Despite of conventional model evaluation metrics, the goodness of a model has to be considered in business context when solving a real-world problem. The model performance has only been marginally improved at the final stage due to the bottleneck of grid searches. Alternatively, furthur optimization can possibly be achieved by random search, re-doing feature engineering, and/or via other machine learning algorithms if time allowed.

4 Reference

Chen, T., He, T., Benesty, M., Khotilovich, V., et al. (2019) R package version 0.82.1. *Xgboost: Extreme gradient boosting.* [Online]. Available from: https://CRAN.R-project.org/package=xgboost.

Kaggle Community (2018) *Steering Wheel of Fortune - Porto Seguro EDA | Kaggle.* [Online]. Available from: https://www.kaggle.com/headsortails/steering-wheel-of-fortune-porto-seguro-eda [Accessed: 18 April 2019].

R Core Team (2018) *R: A language and environment for statistical computing.* [Online]. Vienna, Austria, R Foundation for Statistical Computing. Available from: https://www.R-project.org/.