Porto Seguro's Safe Driver Prediction[*]

Solomon Gebreyohannes

December 20, 2017

Udacity Machine Learning Engineer Nanodegree Capstone Project

Contents

[*]Source: The challenge and data sets are taken from Kaggle competition Kaggle Competition (2017); Kaggle - Data source (2017).

1 Definition

1.1 Project Overview

According to Porto Seguro Porto Seguro (2017), one of Brazil's largest auto and homeowner insurance companies, nothing ruins the thrill of buying a brand-new car more quickly than seeing new insurance bill. The sting's even more painful in case of a good driver. The company believes that it is not fair that one pays so much if he/she is cautious on the road for years. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones. The goal is to utilize machine learning techniques to predict insurance claims with minimum inaccuracy. I propose this challenge entirely taking from Kaggle competition Kaggle Competition (2017), since it is listed as one of the references to get capstone project ideas at Udacity Machine Learning class material. The data needed to train and test the machine learning algorithms are also provided from the same source, Kaggle Kaggle - Data source (2017).

There are several related work on machine learning applications. Scikit-learn Scikit-learn (2017) provides efficient tools for data mining and analysis. Python scripts and documentation (with examples) are available for classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. Panos and Christof summarize machine learning approaches and tools in P. Louridas and C. Ebert (2016). They also present a case study of using machine learning for software code analysis. Tziridis et al. used different machine learning techniques to predict airfare prices Tziridis et al. (2017). Data collection, feature selection, model selection, and evaluation for the case study are discussed in the paper. Using a Kaggle competition challenge, Sangani et al. demonstrate linear regression and gradient boosting to minimize Zillow property estimation error Sangani et al. (2017). Similar to regression, there are classification techniques that predicts a probability distribution over a set of classes instead of outputting where an object or instance belongs to. Garg and Roth discusses probability classifiers in Garg and Roth (2001). This topic is also discussed in Wikipedia Wikipedia-Probabilistic Classification (2017).

1.2 Problem Statement

The challenge is to build a model that predicts the probability that a driver will initiate an auto insurance claim in the next year. This is a prediction problem since it asks a probability, which is a continuous variable over [0,1]. However, the data for training is given in a classification structure, i.e., policy holders (with given features) vs. whether or not they filed an insurance claim last year. Therefore, it is a probabilistic classification problem. A probabilistic classifier predicts a probability distribution over a set of classes, rather than only outputting the most likely class that the observation should belong to Wikipedia-Probabilistic Classification (2017). The features, which will be inputs to the classifier, are grouped into ind, reg, car, and calc. They are also mixed in type - continuous or ordinal, categorical, and binary. Some values of the features are missed from the observation, which are indicated by -1.

The solution is using probabilistic classifiers after preparing the data. It consists of three major parts, viz., data preprocessing, model, and refinement. (1) Data preprocessing: The categorical features (to flatten) and the binary features (to separate the missing values) will be transformed into binary, and missing values (which are represented by -1) will be deleted from the data. The missing values from the continuous features will be predicted using supervised learning technique. Then, the continuous features will be normalized and scaled over [-0.5 0.5]. PCA will be applied on the combined (categorical, binary, and continuous) data to reduce the dimension. The operations will be done on both training and testing data. (2) Model: Probabilistic classifiers will be used to predict the probability of insurance claim. (3) Refinement: After looking on the score of the predicted values, refinement on the model will be done to result a better result. This will be an iterative process. A couple of options explored are using GridSearch and not using PCA.

1.3 Metrics

The result is evaluated using the Gini Coefficient.

Economic Explanation of Gini Coefficient

The Gini coefficient is a measure of inequality of income or wealth. It is usually defined mathematically based on the Lorenz curve, which ranges from 0 to 1 as shown in Figure 1 Wikipedia-Gini coefficient (2017).

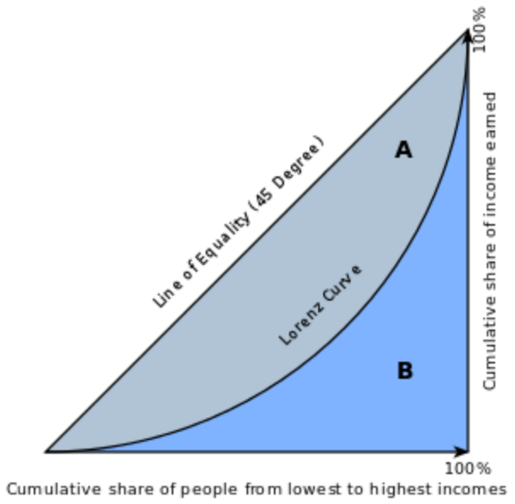

100%
A
Line of Equality (45 Degree)
Lorenz Curve
B
Cumulative share of income earned
100%
Cumulative share of people from lowest to highest incomes

Figure 1: Graphical representation of the Gini coefficient Wikipedia-Gini coefficient (2017)

The Gini coefficient G is calculated as:

$$G = \frac{A}{A+B} \tag{1}$$

where A is the area between the line of equality and the Lorentz curve and $A+B$ is the total area.

Calculating Gini Coefficient for the Project

As explained in [https://www.kaggle.com/batzner/gini-coefficient-an-intuitive-explanation], instead of going through the population from poorest to richest, predictions from lowest to highest will be used and instead of summing up the income as in Figure 1, the actual values of predictions will be summed to find the Gini coefficient for this project.

For example, let the actual and predicted values be [0.0, 0.2, 0.2, 0.6, 0.37, 0.19, 0.3, 0.7, 0.4, 0.8, 0.2, 0.16] and [0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1] respectively. The sorted actual values by predictions are [0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1]. The cummulative actual values are [0, 1, 1, 2, 2, 2, 2, 2, 2, 3, 4, 5]. This is plotted in Figure 2 to show the Lorentz curve.

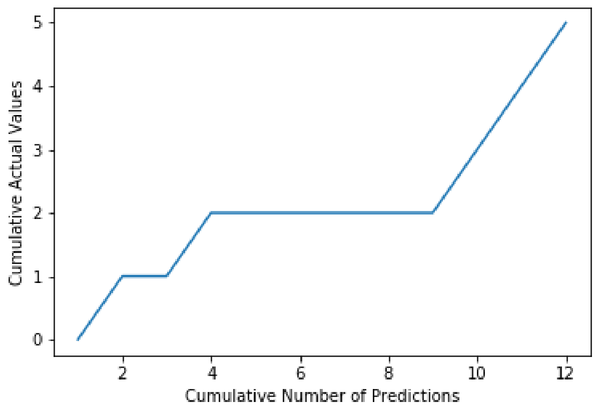


Figure 2: Lorentz curve example

The Gini coefficient is calculated using Equation 1 and it is 0.183. The Gini coefficient ranges from approximately 0 for random guessing, to approximately 0.5 for a perfect score.

For this project, results are evaluated using the Normalized Gini Coefficient, which ranges from 0 (for guessing) to 1 (for perfect score). The scoring algorithm compares the cumulative proportion of positive class observations to a theoretical uniform proportion. The code to calculate Gini Coefficient can be found at Normalized Gini coefficient calculation (code) (2017).

Appropriateness

The Gini coefficient is an appropriate metrics for this challenge. (1) It measures diversity and order of risk (distinguish good from bad cases) The Gini Coefficient (2017), which aligns with the context of this challenge. (2) It sets a score of 0 for random guessing classifier and 1 (for the normalized) for a perfect classifier as required in the challenge.

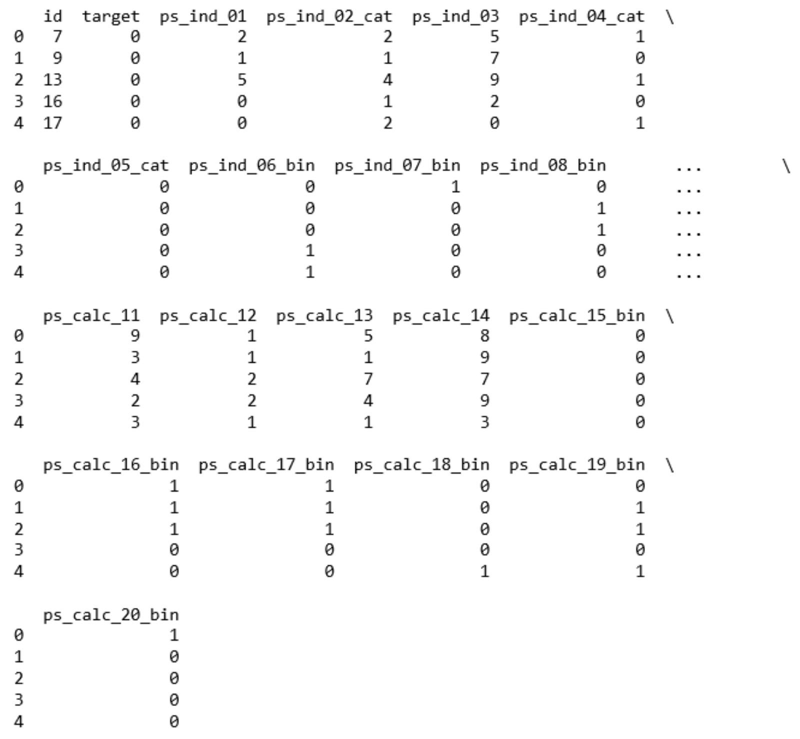
2 Analysis

2.1 Data Exploration

The datasets contain training and testing data. They are taken from Kaggle competition, available at Kaggle - Data source (2017).

The data for training is given in a classification structure, i.e., policy holders (with given features) vs. whether or not they filed an insurance claim last year. The features are grouped into ind, reg, car, and calc. They are also mixed in type continuous or ordinal, categorical, and binary. There 57 different features are (26 continuous features + 14 categorical features + 17 binary features). Some values of the features are missed from the observation, which are indicated by -1.

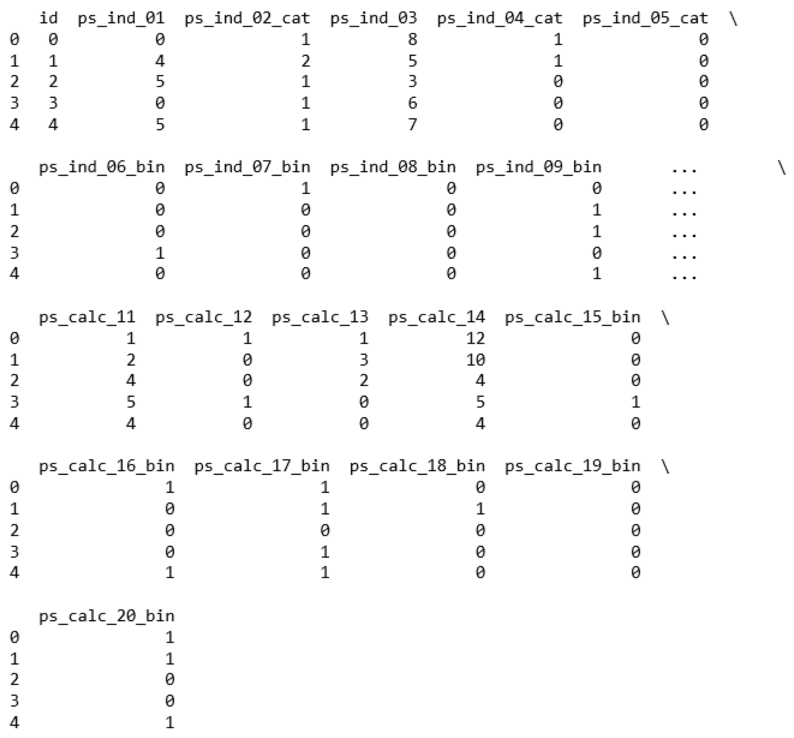
Training and testing data are available in csv form. The training data, `train.csv`, contains policy holders' information vs. whether or not they filed an insurance claim last year. There are 595,212 rows in the training dataset. A sample of the training dataset is shown in Figure 3.



```
   id  target  ps_ind_01  ps_ind_02_cat  ps_ind_03  ps_ind_04_cat  \
0   7       0          2              2          5              1
1   9       0          1              1          7              0
2  13       0          5              4          9              1
3  16       0          0              1          2              0
4  17       0          0              2          0              1

   ps_ind_05_cat  ps_ind_06_bin  ps_ind_07_bin  ps_ind_08_bin       ...        \
0              0              0              1              0       ...
1              0              0              0              1       ...
2              0              0              0              1       ...
3              0              1              0              0       ...
4              0              1              0              0       ...

   ps_calc_11  ps_calc_12  ps_calc_13  ps_calc_14  ps_calc_15_bin  \
0           9           1           5           8               0
1           3           1           1           9               0
2           4           2           7           7               0
3           2           2           4           9               0
4           3           1           1           3               0

   ps_calc_16_bin  ps_calc_17_bin  ps_calc_18_bin  ps_calc_19_bin  \
0               1               1               0               0
1               1               1               0               1
2               1               1               0               1
3               0               0               0               0
4               0               0               1               1

   ps_calc_20_bin
0               1
1               0
2               0
3               0
4               0
```

Figure 3: Training data sample

The test data, `test.csv`, contains only policy holders' information (without the information of insurance claim). There are 892,816 rows in the testing dataset. A sample of the testing dataset is shown in Figure 4.



```
   id  ps_ind_01  ps_ind_02_cat  ps_ind_03  ps_ind_04_cat  ps_ind_05_cat  \
0   0          0              1          8              1              0
1   1          4              2          5              1              0
2   2          5              1          3              0              0
3   3          0              1          6              0              0
4   4          5              1          7              0              0

   ps_ind_06_bin  ps_ind_07_bin  ps_ind_08_bin  ps_ind_09_bin       ...        \
0              0              1              0              0       ...
1              0              0              0              1       ...
2              0              0              0              1       ...
3              1              0              0              0       ...
4              0              0              0              1       ...

   ps_calc_11  ps_calc_12  ps_calc_13  ps_calc_14  ps_calc_15_bin  \
0           1           1           1          12               0
1           2           0           3          10               0
2           4           0           2           4               0
3           5           1           0           5               1
4           4           0           0           4               0

   ps_calc_16_bin  ps_calc_17_bin  ps_calc_18_bin  ps_calc_19_bin  \
0               1               1               0               0
1               0               1               1               0
2               0               0               0               0
3               0               1               0               0
4               1               1               0               0

   ps_calc_20_bin
0               1
1               1
2               0
3               0
4               1
```

Figure 4: Testing data sample

Table 1 summarizes properties of the data set.

Table 1: Summary of properties of the data set

Property	Value
Claims in the training dataset	3.65 % (21694 of 595212)
Non-claims in the training dataset	96.35 % (573518 of 595212)
Training data samples without missing value of any feature	21 % (124931 of 595212)
Testing data samples without missing value of any feature	21 % (186567 of 892816)
Ratio of training size to test size	595212 : 892816 $\approx$ 2:3

2.2 Exploratory Visualization

Categorical Features

Figure 5 shows the count plot of the categorical features. It also shows the relative occurence of the features; this includes the missing values, which are represented by -1. Reading Figure 5, any of the features can not be ignored since they have correlation (not independent) with both “claim” and “not claim” target values.

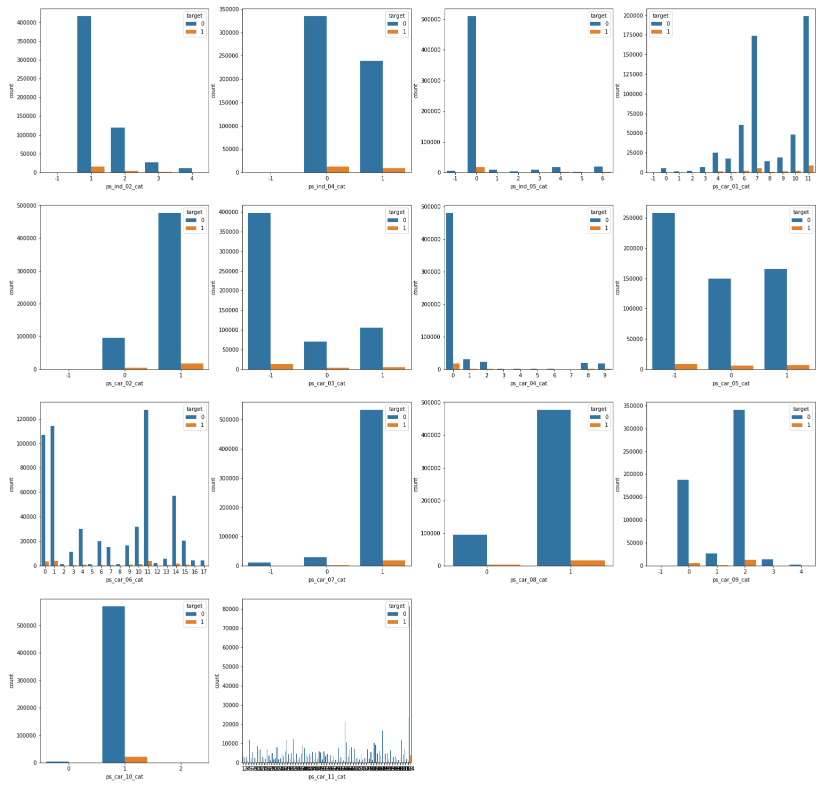


Figure 5: Categorical features

Continuous Features

Figure 6 shows the violin plot of the continuous features vs. target.

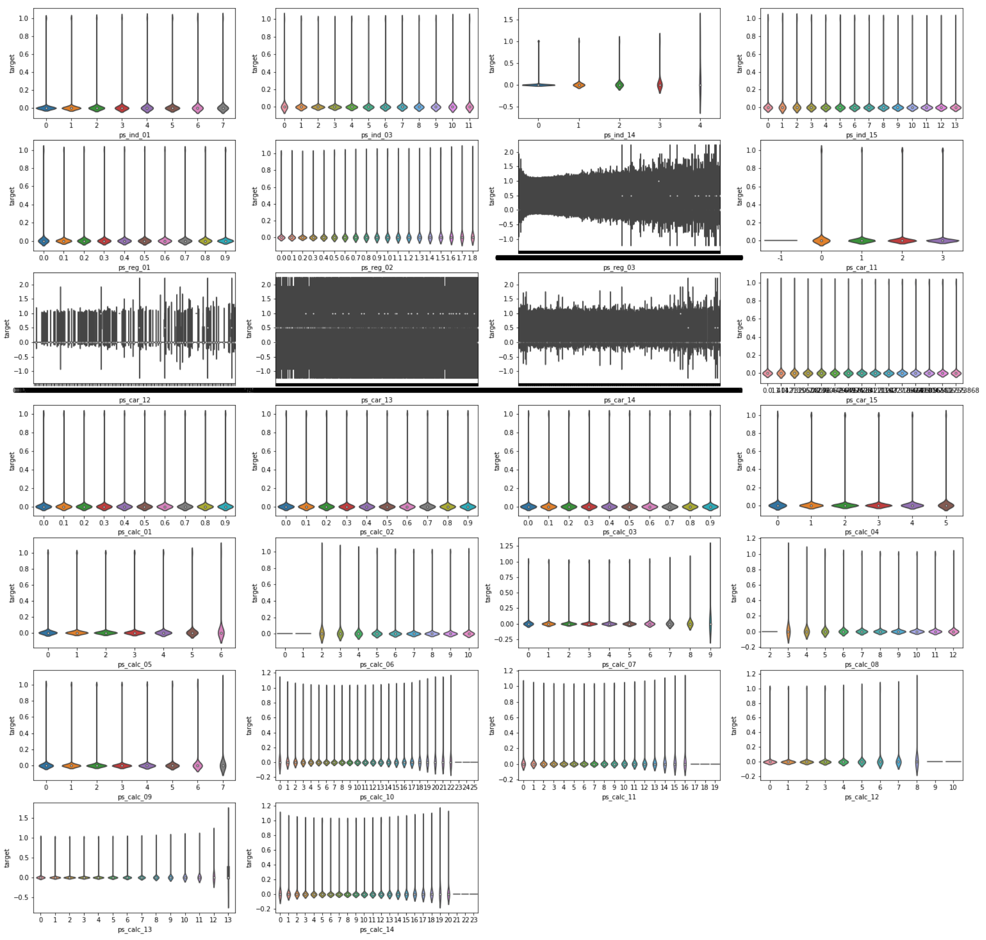


Figure 6: Continuous features violinplot

Combined Features

Figure 7 shows the correlation plot. Reading the correlation plot, the features are not much correlated; therefore, any of the features can not be deleted since its information can not be deduced from the other (i.e., deleting a feature results in an information loss).

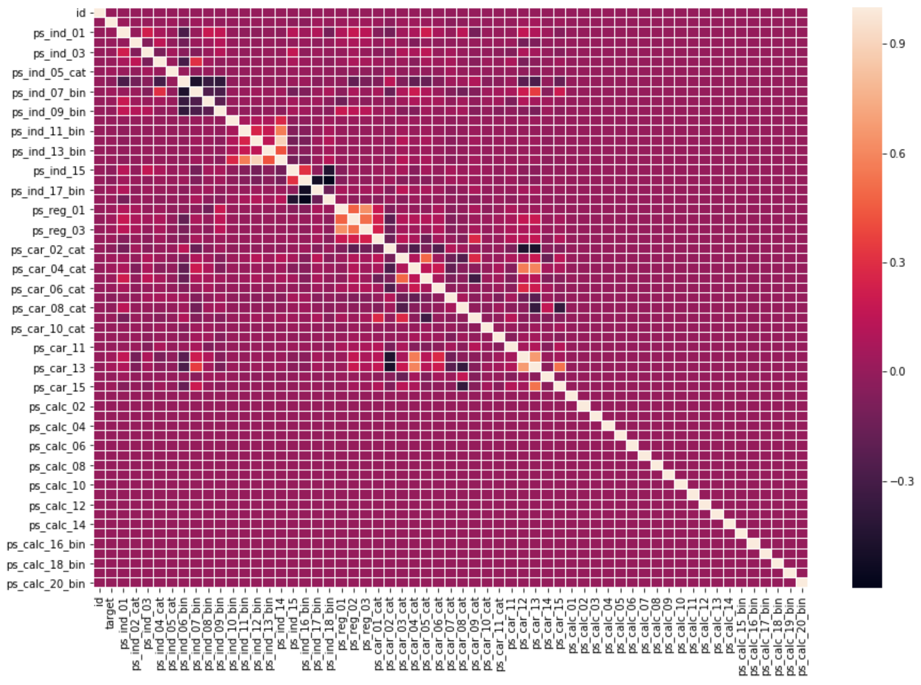


Figure 7: features correlation

2.3 Algorithms and Techniques

Ensemble methods combine the predictions of several base estimators to improve generalizability . With large data like this challenge, ensemble methods are good choices. They have two families, i.e., averaging and boosting methods. In this project, random forest classifier (from averaging methods) is used as a benchmark; and extreme gradient boosting (from boosing methods) is selected as a solution model.

2.3.1 Random Forest Classifier

By taking the notion of "decision trees" (that learns a model by repeatedly splitting subsets of the training examples according to some criteria) to the next level, random forests (as shown in Figure 8) are strong learners Blackwell (2017). They are "a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest" Breiman (2001).

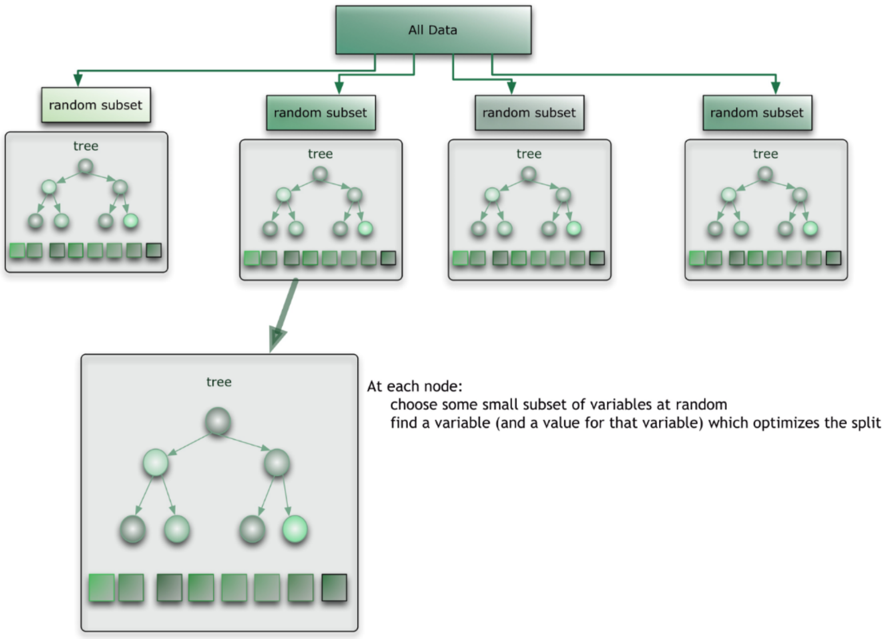

All Data
random subset
random subset
random subset
random subset
tree
tree
tree
tree
tree
At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

Figure 8: Random forest Blackwell (2017)

The random forest classifier results either an average of all the terminal nodes that are reached or a voting majority and is known with its high-accuracy, simple operations, and robustness.

2.3.2 Extreme Gradient Boosting

XGBoost is a supervised learning algorithm that implements a process called boosting to yield accurate models Mitchell and Frank (2017). It is at its core a decision tree boosting algorithm. Boosting refers to the ensemble learning technique of building many models sequentially, with each new model (still a decision tree) attempting to correct for the deficiencies in the previous model.

2.4 Benchmark

Random forest classifier, which averages to improve the predictive accuracy and control over-fitting, is used as a benchmark model. The (Kaggle) Gini score for the benchmark is 0.23739 (Private) and 0.22683 (Public). Table 2 shows the score and parameter setting of the benchmark model.

Table 2: Gini score and parameter setting of random forest (used as benchmark model)

Parameter	Value	(Kaggle) Gini score
n_estimators	100	0.23739 (Private) 0.22683 (Public)
criterion	'gini'	
max_depth	10	
random_state	0	

3 Methodology

3.1 Data Preprocessing

Flattening of the categorical and binary features

The categorical features are tranformed into binary features using one-hot encoding. This will arrange missing values of each feature as a separate feature. To separate the missing values (represented as -1) from the binary features, we also use one-hot encoding for the binary features too.

Missing values of the continuous features

Missing values of the continuous features are replaced by the mean of each column.

Scaling of the continuous features

Scaling of the continuous features is done using `MinMaxScaler`. It scales to a range of [0,1].

Combine all the features

Now, all features will be combined for training and testing data sets.

Dimension reduction using PCA

PCA is used for dimension reduction.

3.2 Implementation

Extreme Gradient boosting classifier is implemented. The (Kaggle) gini score of this model (with PCA) is 0.25192 (Private) and 0.24483 (Public). This result is improved by not using PCA to 0.27345 (Private) and 0.26842 (Public). This is further improved by not including the features without "ps_calc" to a (Kaggle) gini score of 0.27385 (Private) and 0.26915 (Public). Continuing the implementation to improve the score, the next step tried is using GridSearch on XGBoost. However, this is taking forever. Then, I reviewed the Kaggle discussion forums and found one method Johannesss (2017) that implements stacking of LGBClassifiers in to logistic regression. Following this implementation, I implemented stacking XGBClassifiers (instead of LGBClassifiers) in to logistic regression. This results a (Kaggle) gini score of 0.27825 (Private) and 0.27346 (Public).

3.3 Refinement

Refinement is done with no PCA and using XGBClassifiers stacked in a logistic regression. Table 3 summarizes the hyper-parameters used for the models implemented.

Table 3: Summary of models implemented

Model	Parameters
Random forest clasifier	n_estimators = 100, criterion = 'gini', max_depth = 10, random_state = 0
XGBoost	Default parameters
XGBoost, No PCA	Default parameters
XGBoost, no PCA + without "ps_calc" features	Default parameters
XGBoost_Logistic	XGBClassifier(), XGBClassifier(n_estimators=125), XGBClassifier(n_estimators=150) + Logistic regression

Figure 9 presents the (Kaggle) gini score of the models implemented.

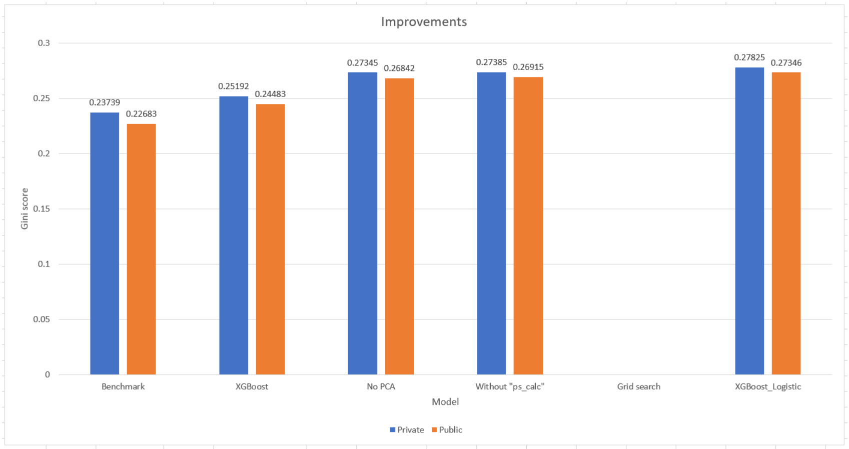
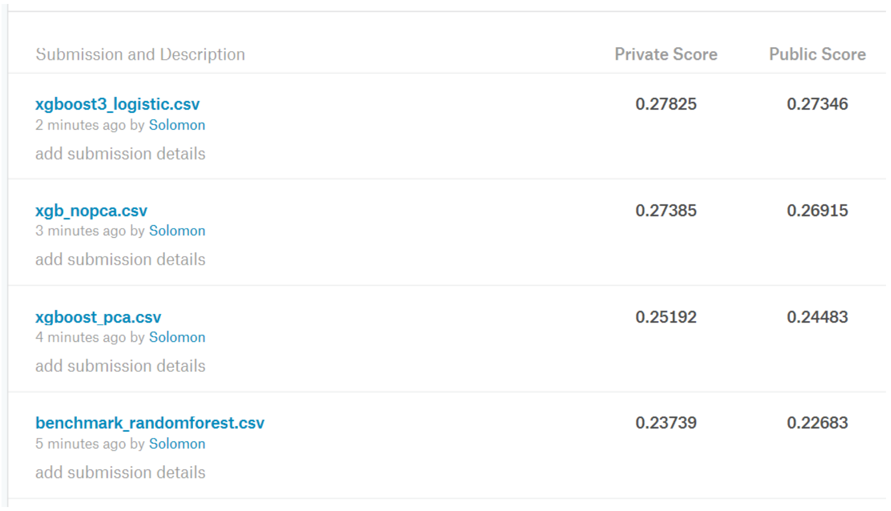


Figure 9: Improvements

4 Results

Figure 10 summarizes the Kaggle submission scores of the different models. Recall that a score of 0 is guessing and 0.5 is a perfect score.



Submission and Description	Private Score	Public Score
xgboost3_logistic.csv 2 minutes ago by Solomon add submission details	0.27825	0.27346
xgb_nopca.csv 3 minutes ago by Solomon add submission details	0.27385	0.26915
xgboost_pca.csv 4 minutes ago by Solomon add submission details	0.25192	0.24483
benchmark_randomforest.csv 5 minutes ago by Solomon add submission details	0.23739	0.22683

Figure 10: Summary of results from Kaggle submission

4.1 Model Evaluation and Validation

Selection of final model

XGBoost is chosen due to a better preliminary result over LGBM and NN MLP classifiers. Then, parameter tuning using grid search was proposed to be done on the Xgboost model. However, because of the large searching time, the final model is changed to three xgboost classifiers stacked in a logistic regression.

Sensitivity analysis

4.2 Justification

The Gini score using the benchmark random forest model is 0.23739 (Kaggle private score) and 0.22683 (Kaggle public score). The Gini score for the final model is 0.27825 (Kaggle private score) and 0.27346 (Kaggle public score). The final model is better than the benchmark by 17.21 % (for private score) and 20.56 % (for public score). Figure 11 shows the comparison between the final model score and the benchmark score.

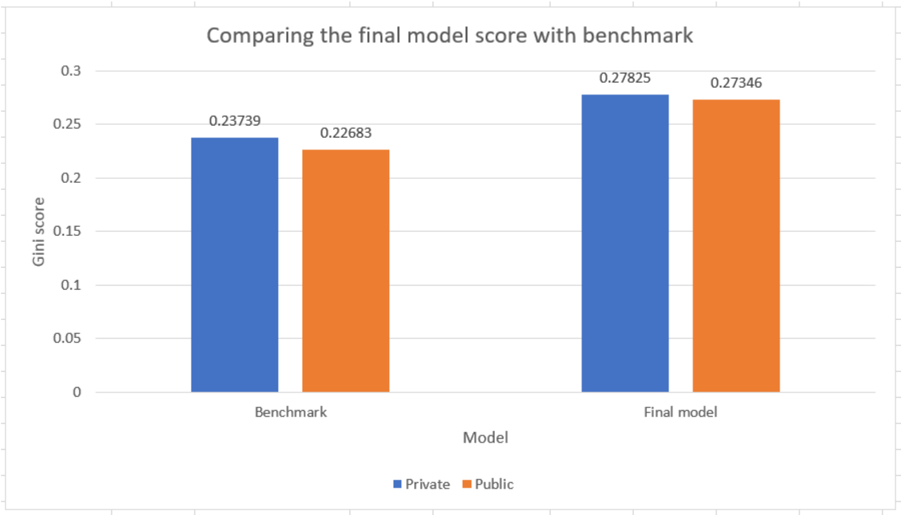


Figure 11: Comparison of final model and benchmark

5 Conclusion

5.1 Free-Form Visualization

Figure 12 shows the feature importance plot.

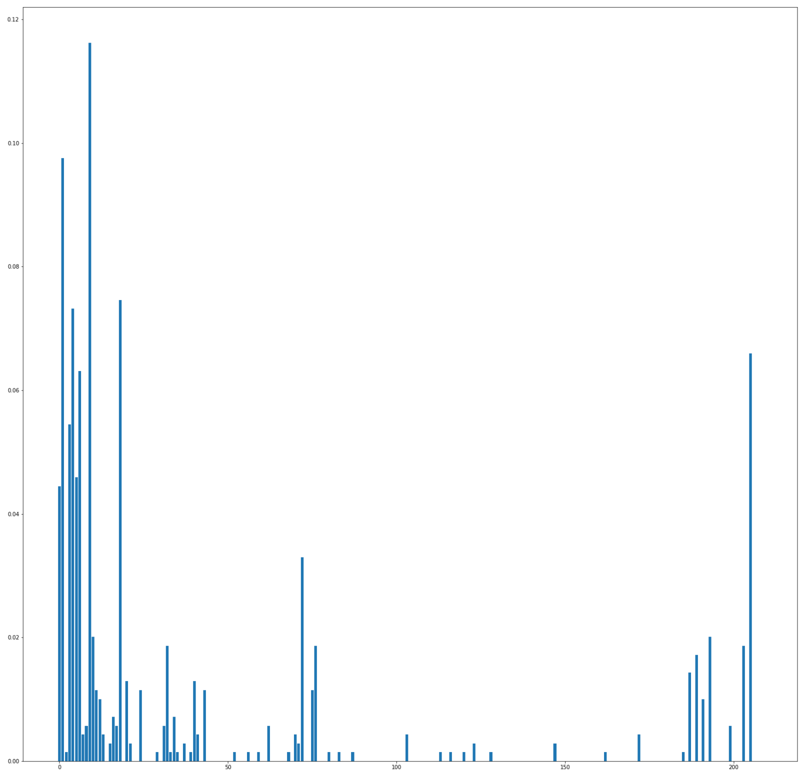


Figure 12: Feature importance plot

5.2 Reflection

This challenge asks the probability of insurance claim with a given training data (people claimed/not claimed and their features) in classification structure, i.e., a probabilistic classification problem. The steps followed to solve this challenge are: (1) Data preprocessing (flattening of categorical features, scaling, feature selection, and dimension reduction) (2) Choose model (random forest for benchmark and xgboost for final model) (3) Refine the model (no PCA, selecting some features, and cross-validation + logistic regression) (4) Report the result (5) Recommend possible extension for better result.

Some observation about this challenge are: (1) Data size - relatively large training and testing datasets. This slows down the grid search algorithm. (2) Score - there is only small score increments on model refinement.

5.3 Improvement

The result might be improved if multiple similar models are used and averaged (or use regressor) over the results. This also help in reducing overfit. Tuning the hyper-parameters more will also help improve the result.

References

Blackwell, A. (2017). Blog: A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System. *http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics*.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Garg, A. and Roth, D. (2001). Understanding probabilistic classifiers. *Machine Learning: ECML 2001*, pages 179–191.

Johannesss (2017). Stacking lightgbm with logisticregression. *https://www.kaggle.com/johannesss/stacking-lightgbm-with-logisticregression*.

Kaggle Competition (2017). *https://www.kaggle.com/c/porto-seguro-safe-driver-prediction*.

Kaggle - Data source (2017). *https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data*.

Mitchell, R. and Frank, E. (2017). Accelerating the xgboost algorithm using gpu computing. *PeerJ Preprints*, 5:e2911v1.

Normalized Gini coefficient calculation (code) (2017). Normalized gini coefficient calculation (code). *https://www.kaggle.com/c/ClaimPredictionChallenge/discussion/703*.

P. Louridas and C. Ebert (2016). Machine learning. *IEEE Software*, 33(5):110–115.

Porto Seguro (2017). *https://www.portoseguro.com.br/*.

Sangani, D., Erickson, K., and Al Hasan, M. (2017). Predicting zillow estimation error using linear regression and gradient boosting. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 530–534. IEEE.

Scikit-learn (2017). Scikit-learn. *http://scikit-learn.org/*.

The Gini Coefficient (2017). The gini coefficient. *http://www.rhinorisk.com/Publications/Gini%20Coefficients.pdf*.

Tziridis, K., Kalampokas, T., Papakostas, G., and Diamantaras, K. (2017). Airfare prices prediction using machine learning techniques. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pages 1036–1039. IEEE.

Wikipedia-Gini coefficient (2017). Gini coefficient. *https://en.wikipedia.org/wiki/Gini_coefficient*.

Wikipedia-Probabilistic Classification (2017). Probabilistic classification. *https://en.wikipedia.org/wiki/Probabilistic_classification*.