Porto Seguro's Safe Driver Prediction

Divyanshu Sharma
MT2020061
International Institute of Information Technology
Bangalore
divyanshu.sharma@iiitb.org

Mitisha Agrawal
MT2020123
International Institute of Information Technology
Bangalore
mitisha.agrawal@iiitb.org

Neeraj Jetha
MT2020079
International Institute of Information Technology
Bangalore
neeraj.jetha@iiitb.org

***Abstract*—In the recent scenarios many automotive insurance providers are looking to improve their service for their customers. Businesses are starting to adapt and implement Machine Learning and artificial intelligence methods for analysing data for performance, as a result giving better service for their customers from a better understanding of their needs.**

We are presenting a model using traditional machine learning methods to predict the probability that a driver will initiate an auto insurance claim in the next year. Better predictions increase car-ownership accessibility for safer drivers and allow car insurance companies to charge fair prices to all customers. Better predictions also lead to improved profits for insurance companies.

***Index Terms*—Logistic Regression, Random Forest Classifier, Light Gradient Boosting Method(LGBM), AdaBoost, XGBoost, CatBoost, Stacking.**

I. Introduction

Improving the accuracy of insurance claims benefits both customers and insurance companies. Incorrect predictions effectively raise insurance costs for safe drivers and lower costs for risky drivers, and can be costly to insurance companies. Insurance claim occurs when the policyholder (customer) creates a formal request to an insurer for coverage for an accident. The insurance company must validate this request and then decide whether or not to issue payment to policyholder.

Nothing ruins the thrill of buying a brand new car more quickly than seeing your new insurance bill. The sting's even more painful when you know you're a good driver. It does not seem fair that you have to pay so much if you have been cautious on the road for years.

Porto Seguro, one of Brazil's largest auto and homeowner insurance companies, completely agrees. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones.

Vehicle owner seek out automotive insurance companies for insurance so that in the unfortunate event of an accident, they can mitigate the costs involved with coverage for: damage, liability and medical injuries.[1]

The problem is as follows: given a series of unlabeled features collected by an insurance company about a customer, can we predict whether the customer will file an insurance claim during a period of interest?

So, there needs to be an efficient method that can determine the risk a driver poses to a company, in order for the firms to be able to adjust the insurance prices fairly to a drivers ability and relevant personal information making automotive insurance more accessible to drivers , but also consider the insurance firm not losing money.

The overall scope for this project focuses on creating a machine learning algorithm which will provide an automotive insurance predictions as accurately as possible, in order for the client Porto Seguro to be able to improve their business by understanding their customers information in more depth. Given an accurate insurance prediction it will support Porto Seguro in tailoring insurance quotes according to the drivers ability. The model should therefore make it clearer which drivers are unlikely to make a claim and adjusts their insurance quote lower as well as accordingly increase the insurance cost for those who are likely to make a claim.

This report presents various models using traditional machine learning methods to predict the probability that a driver will initiate an auto insurance claim in the next year.

The rest of the paper proceeds as follows: **Sec 2** Discusses related work on the models and techniques that have been used to predict the probability of malware infection. **Sec 3** We describe our dataset. **Sec 4** Covers our visualizations,EDA and their inferences. **Sec 5** will discuss data preprocessing. **Sec 6** will cover the model-building, their implementation and evaluation. **Sec 7** will detail the comparison of all models applied with the best possible solution. **Sec 8** we include training details that we used. We conclude in **Sec 9** and in **Sec 10** we outline challenges with our method and possible avenues of future work.

II. Related Work

Porto Seguro is a company which operates in many lines of insurance including : automobile , business, company health, investments, financing, residence services and telecommunications.

Smith et al. (2000) use several techniques such as decision trees and neural networks to study which customer characteristics affect retention in their policy.

Yeo et al. (2001) employ clustering techniques to group customers according to their risk classification and then regression methods to model the expected claim costs within a risk group.

Outside of automobile insurance, researchers have also studied health insurance (Browne 1992), non-life insurance (Salcedo-Sanz et al. 2005), and many other situations where risk plays a substantial factor in pricing (Lee Urrutia 1996, Cummins Phillips 1999).[2]

III. Dataset

The train datasets consisted of 59 variables including 'ID' and 'Target'. So there are a total of 57 features in the datasets which can be used for prediction. All of the 57 features are kept anonymous for privacy reasons.

In train data set their are 416648 data points , 59 columns and test data set contains 178564 data points , 58 columns

In the train and test data, features that belong to similar groupings are tagged as such in the feature names (e.g., 'ind', 'reg', 'car', 'calc'). In addition, feature names include the post fix 'bin' to indicate binary features and 'cat' to indicate categorical features. Features without these designations are either continuous or ordinal. Values of '-1' indicate that the feature was missing from the observation. The 'target' columns signifies whether or not a claim was filed for that policy holder[3].

Using the information and label in train Datasets we're challenged to build a model that predicts the probability of each data points in test Datasets that a driver will initiate an auto insurance claim in the next year.

IV. Observation

Before getting into preprocessing and feature extraction, it is very important to get to know the distribution of data in order to get better insights for feature selection.The problem is Classification to predict whether the data point belongs to class 0 or 1
0→ No claim for insurance
1→ claim for insurance

- The dataset is imbalanced, having 401427 person who have not claimed for insurance and 15221 person who claimed for insurance.

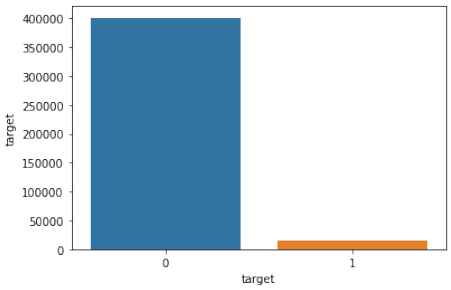

Fig. 1. Target Feature.

- From the data information it is understood that all the features are integers and floats. In the data description it is described that features contain categorical and binary. So we classified data into 3 classes as Binary, Categorical and Numeric features.

TABLE I
Classes

Class	*Count*
Binary Features	17
Categorical Features	12
Numeric Features	26

- Their are null values in the features which need to be removed first from the dataset .The feature which has more than 50% null values need to dropped.Their are features which has high percentage of null values.

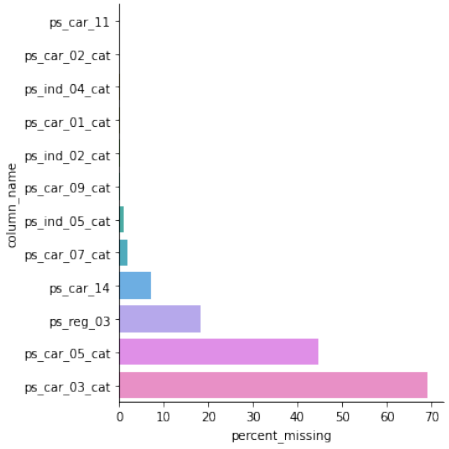

Fig. 2. Feature containing Null values

- There are a strong correlations between the features. Fig.3.
- There are a strong negative correlation between the binary features .Fig.4

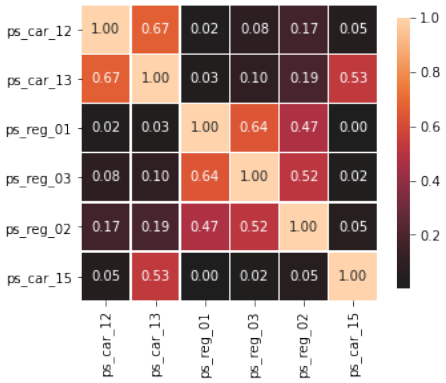

Fig. 3. Features with High correlation

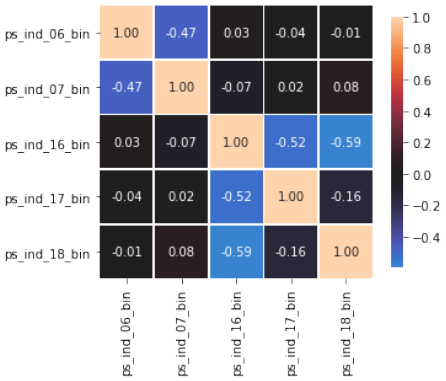

Fig. 4. Defender

V. DataPreprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.While going through the dataset, we observed that data was not clean and formatted. There was need for cleaning the data and making it suitable for a model fitting which will also help in increasing the accuracy and efficiency of a machine learning model. Hence, preprocessing the data was a very crucial step in training our model.

The dataset contained many features which had NULL values.The features which had more than 50% NULL values did not contribute much to the target feature. We observed that ps_car_03_cat and ps_car_05_cat have a large proportion of records with missing values, so we removed these features from train as well as test dataset.The features which had less than 50% NULL values for all of them, we replaced their NULL values with meanvalues both in train and test dataset.We divided the features into three classes which were numerical feature , categorical feature and binary feature.

Binary features contains two values 0 and 1 . In the dataset feature name contains bin as postfix so they are binary features. Categorical features contains the numerical value whose dtype=integer these feature are identified as they contain cat as postfix in the feature name and the remaining features are numeric features.

Machine learning models can only deal with numbers. Often times our data contains numerical as well as categorical data. So the categorical data must be properly encoded to feed into a ML model. There are various methods of categorical encoding such as One-Hot-Encoding[4], label-encoding etc. It has to be through experimentation that we should determine which encoding technique works the best for our models.We tried combinations of these encoding techniques with the models applied. All the above mentioned techniques were tried for this problem and it was found that Label encoding was the one performing the best.

VI. Model Selection

After going through EDA we analyze the data and than Data Processing is done now the data is clean it is ready to be fed into the model.A model has to be hypertuned properly to get the best result .Sometimes the complicated powerful model may b be required for the problem but the sometimes low model can do the better job. The model selection can be better understood by experimentation.

All the hyper-parameter tuning for the below models was done manually because of the computational overheads of GridSearchCV or RandomSearchCV. Also manual tuning gave more insight on how the model behaved with small changes in the parameters.

We executed some classical machine learning models like Logistic regression, AdaBoost, RandomForest Classifier, XGBoost, LGBM, CatBoost, and Stacking of these models. Since our dataset was unbalanced we tried to balance the dataset using SMOTE (synthetic minority oversampling technique) algorithm but the result/score that we obtained after applying oversampling was not better,than the one we got without oversampling. So we discarded oversampling and applied models on unbalanced dataset.

A. Logistic Regression

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. Logistic regression has become an important tool in the discipline of machine learning. The approach allows an algorithm being used in a machine learning application to classify incoming data based on historical data. As more relevant data comes in, the algorithm should get better at predicting classifications within data sets.

Before Applying the logistic regression we found the important 25 features from the dataset. For this we used the Random Forest algorithm for feature importance

implemented in scikit-learn as the RandomForestRegressor and RandomForestClassifier classes.

Logistic Regression is applied on the top 25 important features that we extracted.[5]

Implementation and Evaluation:The Normalized Gini Index on the public leaderboard that we obtain using logistic regression is 0.26540 and the one on private leader board is 0.23127 which was one of the lowest score.We also tried to balanced the dataset but the Gini coefficient came very low.

B. Random Forest

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Implementation and Evaluation:While applying the random forest classifier we tried on different sets of the hyperparameters to get better results.We got better results than logistic regression but we could do limited changes in the hyperparameters as when we tried increasing the n_estimators the model became slower and slower, so, The best Gini Index on the public leaderboard that we could get from random forest classifier is 0.26567 and on private leaderboard 0.23910 was the gini index.The time required to train the data was nearly 3hrs.

C. AdaBoost

AdaBoost helps us to combine multiple "weak classifiers" into a single "strong classifier" .The weak learners in AdaBoost are decision trees with a single split, called decision stumps. AdaBoost works by putting more weight on difficult to classify instances and less on those already handled well. AdaBoost algorithms can be used for both classification and regression problem.

Implementation and Evaluation:We splitted the train data into train and validation in the ratio of 80/20 while creating the model.We used the base as DecisionTreeClassifier while working on Adaboost.[7] We also tried using the SVC as base estimator[8] but we could not run it because of the time duration restrictions on online platforms. The Gini Index for the AdaBoost algorithm is 0.26199 on public leaderboard and 0.24300 on private leaderboard. Adaboost out performed Random Forest Classifier on the private leaderboard whereas it scored less in the public leaderboard. The time required to completely train the data is nearly 6 hrs.

D. XGBoost

XGBoost stands for Extreme Gradient Boosting Algorithm .The XGBoost algorithm was implemented to maximize the efficiency of compute time and memory resources. The algorithm features Sparse Aware implementation by automatically handling missing values from data sets. Additionally, the algorithm allows for continued training of an already fitted model on new data.

Implementation and Evaluation:We splitted the Train data into train and validation in the ratio of 80/20 while creating the model.We tried different combinations of hyperparameters[9] while working with Xgboost and we got better results than the other models we tried so far.The Gini Index for the XGBoost algorithm on validation dataset is 0.27789 but we observed that the model was overfitting and hence was not giving the best possible result.

So, we read about regularization,pruning and other methods[10] to reduce overfit and updated the hyperparameters accordingly, even used the early stopping rounds to estimate the best possible hyperparameters and we could eventually reach the Gini Index of 0.29713 on the public leaderboard and 0.26625 on the private leaderboard, which is our best score for public leaderboard but it is not our best score of private leaderboard.The total time required to train the dataset is nearly 1.2hrs.

E. Light Gradient Boosting Method (LGBM)

Gradient boosting utilizes tree-based learning and its base classifier is based on decision trees. Unlike most traditional boosting methods whereby decision trees are grown breadth-wise, LGBM grows leaf-wise. Growing leaf-wise usually results in a lower loss, but it may cause overfitting when data size is small.

LGBM can handle categorical features by just taking the input of the feature names. One of the many advantages of LGBM is that it can handle large size dataset and takes a low memory to run. It also provides a better accuracy as compared to other boosting frameworks.

Implementation and Evaluation:Initially a 5-fold cross validation with a train valid split of 80/20 was used when creating the model. The folds preserved the percentage of samples for each class using stratified kfold. Simply, it helped to ensure that target relative class frequencies was approximately preserved in each train and validation fold.

The Gini Index for the LGBM model on validation dataset is 0.29002 on public leaderboard and 0.26728 in private leaderboard.After the result declaration on Kaggle we realised that this was our best score on the private leader board but we did not use this as our final submission as on our public leaderboard we had better scores with other models.

Then we tried using the standard LGBM Classifier[11] with some hyperparameter tuning and even ensuring that we

use parameters which might help in not overfitting as we learned in our previous experience of XgBoost model and eventually we reached the Gini Index of 0.29097 on public leaderboard and 0.26482 on private leaderboard. This model performed less than the other variant of LGBM in the private leaderboard The total time required to train the dataset is 45 minutes.

F. CatBoost

"CatBoost" name comes from two words "Category" and "Boosting". CatBoost is a recently open-sourced machine learning algorithm from Yandex

It is especially powerful in two ways:

- It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
- Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems[9].

Implementation and Evaluation:When applying this model their was no requirement of converting the categorical data to numerical data so we did not perform Label Encoding. The hyperparameters which we tried gave us the best result when we used the iteration as 2000 which we obtained after trying some combinations of parameters.Train data was split into train and validation in the 80/20 while creating the model[12].

The Gini Index for the CatBoost algorithm came as 0.28608 in the public leaderboard and 0.26533 in private leaderboard. The time required to completely train the data is nearly 2.5hrs.

G. Stacking

The overall idea of stacking is to train several models, usually with different algorithm types (aka base-learners), on the train data, and then rather than picking the best model, all the models are aggregated/fronted using another model (meta learner), to make the final prediction. The inputs for the meta-learner is the prediction outputs of the base-learners.

Implementation and Evaluation:We used the StackingClassifier available, to stack different models on different levels and try to give better prediction by proving the output of base learners to final estimator. After optimizing different models we used certain combinations of models[13].

- We first tried a combination where we used Xgboost, LGBM model as base estimators and Logistic Regression as final estimator. Using this we could get a Gini Index of 0.29090 on public leaderboard and 0.26710 on private leaderboard.
- Then we tried a combination where we used Xgboost, LGBM and Catboost models as base estimators and Logistic Regression as final estimator. Using this we could get a Gini Index of 0.29105 on public leaderboard and 0.26713 on private leaderboard. Which was our highest score in case of stacking.
- Then we also tried using Catboost and LGBM as base estimator and Xgboost as final estimator, using this we could get a Gini Index of 0.28276 on public leaderboard and 0.25565 on private leaderboard.
  The biggest challenge was the high computational time required in running the StackingClassifier so we used the GPU available in Kaggle to speed up our work by adding hyperparameters to Xgboost and Catboost model we could get faster results.

VII. Result Analysis

We applied the classical models Logistic Regression, Random Forest, Adaboost, XGBoost, LGBM, CatBoost and Stacking with combinations of encoding like One-hot encoding and label encoding.But we obtained better results with label encoding.We obtained our best score on public leaderboard from Xgboost model with Gini Index of 0.29713 and Stacking where we used Base estimators as Xgboost, LGBM and catboost and final estimator was Logistic Regression with Gini Index of 0.29105 . We submitted these two entries as our final submission on Kaggle.

After the declaration of the result of Kaggle Competition we observed that our Stacking submission outperformed the Xgboost model with as score of 0.26173 on private leaderboard.We also saw that our LGBM model performed more well on the private leaderboard with score of 0.26728 but as we did not choose it for final submission it did not contribute towards the final results.

TABLE II
Experiment Results

Experiment Title	*Public Score*	*Private Score*
Logistic Regression	0.26540	0.23127
Random Forest	0.26567	0.23910
AdaBoost	0.26199	0.24300
XGBoost	0.29173	0.26625
LGBM with K-fold	0.29002	0.26728
LGBMClassifier	0.29097	0.26482
CatBoost	0.28608	0.26533
Stacking	0.29105	0.26713

VIII. Training Details

We trained our model on the 80 percent of the data first and analysed the Gini Index if we saw any improvement in our scores then we trained the model on the complete train data this helped us in saving our time and we could work efficiently. We also used the GPU available to us in the

Kaggle platform to speed up our process.
We also used gc.collect() method to call the garbage collector after running the classifier so that unallocated data can be freed. This was useful when we were running multiple classifiers consecutively.

IX. Conclusion

We would like to conclude that we were able to come up with a Xgboost model that gave us the score of 0.29173 on public leaderboard and also with a Stacking model which outperformed the Xgboost model in the private leaderboard with gini index of 0.26713 while predicting the probability that a driver will initiate an auto insurance claim in the next year.

Such projects have a great scope to increase car-ownership accessibility for safer drivers and allow car insurance companies to charge fair prices to all customers.

X. Challenges And Future Scope

Identification of insignificant features is a challenge as we need to remove them before we train our model to obtain better results and with a huge dataset its is difficult to analyse the features.For every insignificant feature input to the classifier, the amount of data needed to accurately generalize the classifier increases exponentially and the training dataset density will decrease exponentially. Hence, by having a large number of input variables to the classifier which is noisy and less correlated, we run the risk of over-fitting. This will result in the classifier being affected by noise in the data and cause our classifier to be less accurate.

Long computational time was also a challenge so to overcome these problems we removed the features that we analysed during preprocessing and we also used the GPU options available.

With the help of more balanced data we could have obtained better results as by applying external methods did not suffice the need.Moreover using neural networks we could achieve better results.

XI. Acknowledgment

We would like to thank Professor G. Srinivas Raghavan, Professor Neelam Sinha and our Machine learning Teaching Assistants for giving us the opportunity to work on the project and help us whenever we were stuck by giving us ideas and resources to learn from.We would like to thank all the teams on our leaderboard to provide us with healthy competition and give us that desire to work harder and perform better by setting new benchmarks everyday.

We would gladly say that we had a great learning experience while working on this project.Having a leaderboard was a great driving fuel to work hard and strive for better results everyday by reading more and more articles and implementing various models with lots of combinations.

References

[1] URL:https://medium.com/@madanchowdary1996/porto-seguros-safe-driver-prediction-machine-learning-case-study-ebbd9a733361
[2] Matthew Millican, Laura Zhang and Dixee Kimball "CS 229 Final Report: Predicting Insurance Claims in Brazil" ,2017. URL:http://cs229.stanford.edu/proj2017/final-reports/5244182.pdf
[3] URL:https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data
[4] URL:https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/
[5] Building A Logistic Regression in Python, Step by Step https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8
[6] URL:https://machinelearningmastery.com/calculate-feature-importance-with-python/
[7] URL:https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html/
[8] URL:https://www.datacamp.com/community/tutorials/adaboost-classifier-python
[9] URL:https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/
[10] URL:https://www.capitalone.com/tech/machine-learning/how-to-control-your-xgboost-model/
[11] URL:https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html/
[12] URL:https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/
[13] URL:https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html/