Porto Seguro's Safe Driver Prediction

Abhishek Verma
42010520001
verma94sunny@gmail.com

***Abstract*—Car insurance can help protect someone from huge expenses. By investing a small amount now in one's own protection, one can help avoid costly expenses in the future. Paying the right amount is fine, but sometimes seeing the price of your insurance bill rocketing through the roof may seem an overkill, knowing well how good a driver you are that you might never need such huge bills.**

So, we are trying to understand how good a driver is, whether he/she will initiate an auto insurance claim in the next year, which will help companies accordingly predict the correct cost of insurance for a driver based on it's driver skill.

***Index Terms*—Feature Engineering, Cross Validation, Logistic Regression, LightGBM, Decision Tree, Random Forest Classifier, SMOTE, XG-Boost.**

Introduction

Having an appropriate insurance for the vehicle can help the owner to be protected from all types of liabilities that could emerge from accidents, which might cause harm to the car, property or for that matter any individual. Also, it is mandatory by the law to have an insurance. But, sometimes the insurance bill is high enough to ruin the thrill of buying a new car. The sting's even more painful when you know you're a good driver, so paying a hefty amount seems unfair, especially when you know that you are a safe driver and have been cautious on the road for years.

Porto Seguro, the leader on the auto and homeowner insurance segments in Brazil which has around 10 million clients all over the different business lines, completely agrees with this situation faced by the drivers. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones.

This report explores the challenges faced using the traditional machine learning methods, in order to overcome this unfair price means, by building an appropriate model that predicts the probability of a driver claiming an auto insurance in the next year, thus knowing how often a driver performs on the road in order to compensate for his or her insurance charges. Porto Seguro has been using machine learning for the past 20 years, and accurate predictions will further allow them to tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

Dataset

The data provided in the dataset contains various features that are grouped together in similar groupings with tags along with it to identify the same for e.g. ind, reg, car, calc, etc. In addition to this, feature names include the postfix 'bin' to indicate binary features and 'cat' to indicate categorical features.

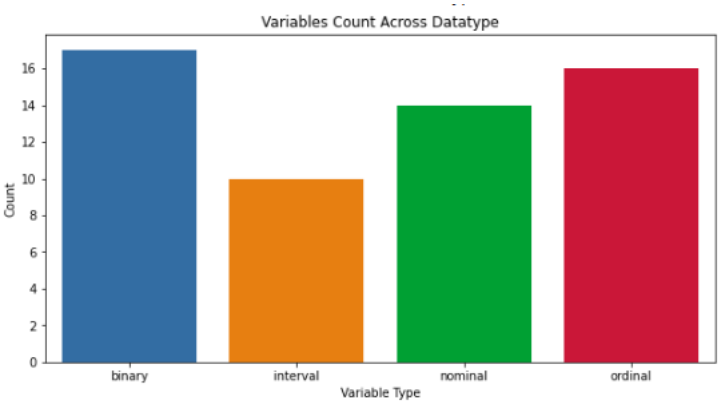
Features without these designations are either continuous or ordinal. Values of -1 indicate that the feature was missing from the observation. The 'target' columns signifies whether or not a claim was filed for that policy holder.

The training dataset consists of 59 variables including 'id' and 'target' making a total of 57 features in the dataset which can be used for prediction. All of the 57 features are sort of unknown and anonymous for some reason, providing no description of what these are making it a bit hard to make sense of them.

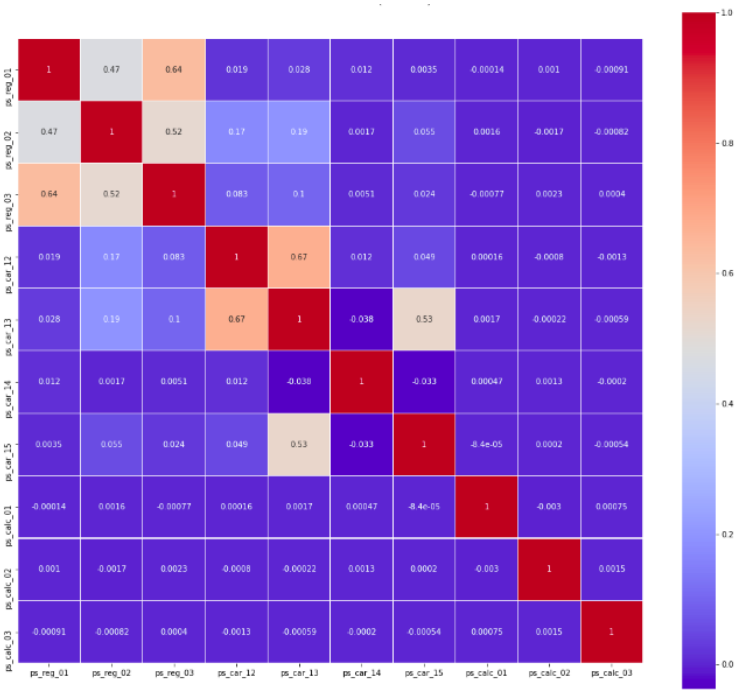
I. Data Visualization

All of the dataset consists of numerical type of features only of type either int64 or float64. The only telling of some features being categorical is by a mere 'cat' label at the end of their names even though the data in them is numerical.

1) We identified different variables and categorised them into four main types: binary, interval, nominal and ordinal, the following graph shows the count of them.



2) Correlation of continuous (interval) features



The categories need to be considered are: ps_reg_01 and ps_reg_03; ps_car_12 and ps_car_13; ps_car_13 and ps_car_15; ps_reg_02 and ps_reg_03;

3) Some of the columns that are supposed to be categorical having categories ≤ 8 are visualized as:

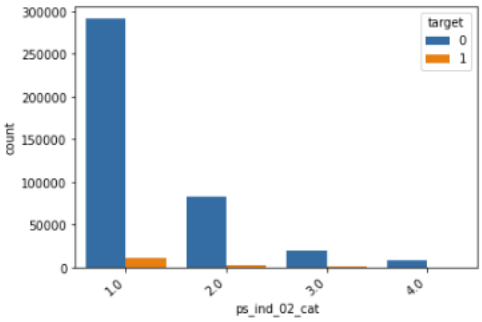
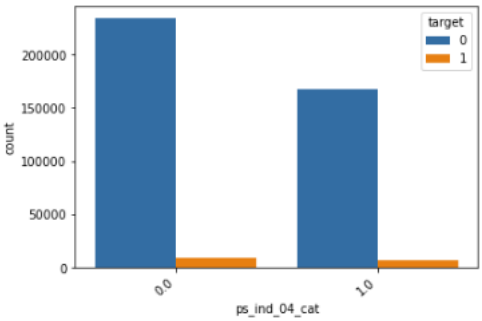
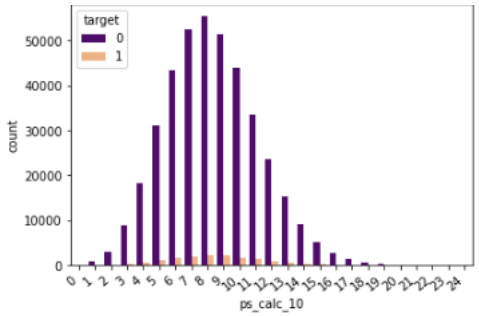
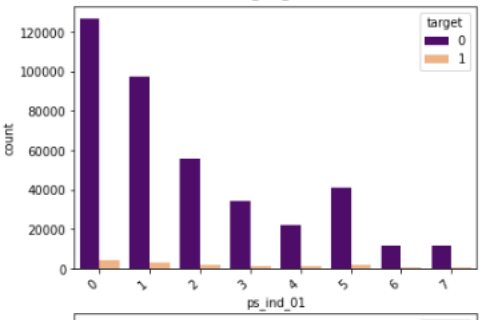


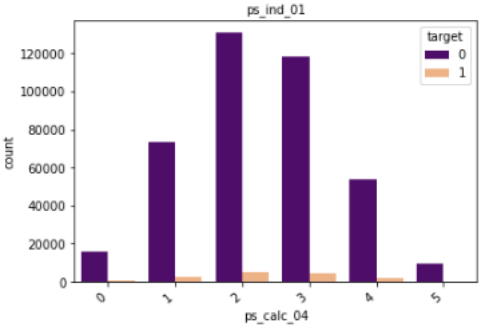
Fig. 1. count of ps_ind_02_cat



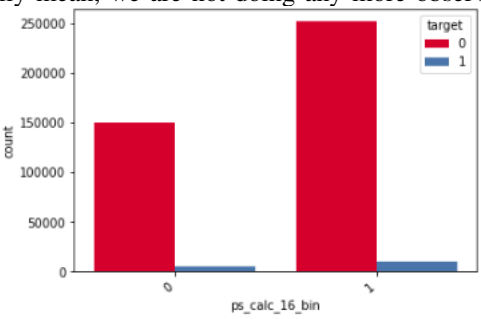
4) Similarly, visualizing some ordinal features

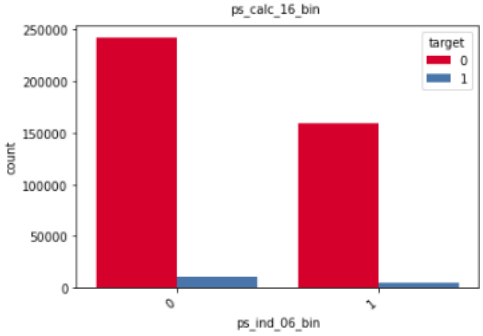






5) Plotting counts of some binary features, and since we can't make much sense of the data given that they are anonymized and there is no clear way of knowing what the features specifically mean, we are not doing any more observations.





II. Data Preprocessing and Feature Engineering

We are given that the missing values are indicated by -1 . So, we replaced -1 with NaN in order to compute how many missing values are present. While looking for missing values, we found that two of the features ps_car_03_cat and ps_car_05_cat had highest missing values being more than 69% and 44% respectviely. So, we removed these features from the dataset.

Further observation showed that we have:

- Categorical missing features (ending with 'cat')
- Continuous missing features (features having numerical continuous values - 'ps_reg_03','ps_car_14','ps_car_12')
- Ordinal missing feature (which is neither continuous nor categorical - 'ps_car_11')

We handled these missing values using regression models - linear regression for continuous features and logistic regression for categorical features. The reason for prediction rather than imputation is that imputing artificial values results in poor accuracy.

For filling in these missing values, we used mean strategy of imputation by SimpleImputer, a scikit-learn class, and for this, we already filled -1 values in the dataset with NaN.

Furthermore, we didn't encode the so called categorical features because all of the data was numerical only, so no need to it.

Upon further examining, there was a heavy class imbalance feature at hand with the target column which specified whether a customer has filed for an insurance claim or not and only around 4% of the total customers have had filed the claim and rest 96% haven't. To handle this imbalance we tried using SMOTE(Synthetic Minority Over-sampling Technique), an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the over fitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

III. Models

Now comes the model building part where we first started off with logistic regression which is appropriate to conduct when the dependent variable is dichotomous (binary) and here it is suitable for predicting an insurance claim which have to variables (1 for yes and 0 for no). The result that logistic regression gave on over sampled data with smote was very poor so we trained it again on the data without applying smote which gave comparatively better results.

The next major model we used was Decision Trees, which are used for both classification and regression tasks.The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data). Decision trees are based off the concept of having a choice of decisions and then following the path of subsequent actions as a result. Following are the advantages of decision trees:-

- Can handle both categorical and numerical data.
- Resistant to outliers, hence require little data preprocessing.
- Can be used to build larger classifiers by using ensemble methods.

Following are the disadvantages of decision trees:-

- Prone to overfitting.
- Can create biased learned trees if some classes dominate.
- Require some kind of measurement as to how well they are doing.

We fed decision tree with both oversampled data and the original data and the later outperformed and gave better results.

The next model we tried was Random Forest Classifier 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. Simply, it builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random forest adds additional randomness to the model, while growing the trees, instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Once again, we didn't use smote because of poor performance and without smote, we got a score of 0.26178 which was an improvement.

The hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster. The one's we used that gave us better performance were:

- n_estimators : the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions
- random_state : makes the model's output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same hyperparameters and the same training data.
- n_jobs : tells the engine how many processors it is allowed to use. If it has a value of one, it can only use one processor. A value of "-1" means that there is no limit.
- max_depth : the maximum depth of the tree
- min_samples_split : minimum number of samples required to split an internal node.
- min_samples_leaf : minimum number of samples required to be at a leaf node.

The next model we used was XGBoost, it gave us the best result with smote and it's score was highest amongst all the previous models. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that carries out the gradient boosting decision tree algorithm. Gradient boosting is a method where the new models are created that computes the error in the previous model and then leftover is added to make the final prediction. The advantage of this iterative approach is that the new models being added are focused on correcting the mistakes which were caused by other models. In

a standard ensemble method where models are trained in isolation, all of the models might simply end up making the same mistakes. Getting good results with XGBoost also depends on hyperparameter tuning and the parameters we used were : learning_rate, n_estimators, max_depth, gamma, subsample, colsample_bytree, objective, reg_alpha, reg_lambda, n_jobs, seed. We tried different tunings with these hyperparameters, and surprisingly xgboost's result were better with oversampled data with smote rather than without smote as was the case with the previous models.

Further we tried a little bit of simple ensembling methods like : max voting algorithm, weighted averaging and power averaging. In this, we included another model - GaussianNB.

The Naive Bayes classifiers working is based on the Bayes' theorem, which describes the probability of an event, based on prior knowledge of conditions be related of conditions to the event.

Now coming to simple ensembling, we used four models for this namely - logistic regression, gaussian naive bayes, decission tree classifier and random forest classifier.

The results of max voting were not an improvement over our best result but the weighted average technique although not the best was somewhat better. Instead of just multiplying the weights to the different model predictions and adding them like :

$$m_1 \times w_1 + m_2 \times w_2 + m_3 \times w_3 + m_4 \times w_4$$

we also tried to modify or optimise the results by taking power of these weights to the model predictions and averaging them like :

$$(m_1^{w_1} + m_2^{w_2} + m_3^{w_3} + m_4^{w_4})/4$$

Although there was not much of a noticeable difference in scores.

And lastly, the power averaging method did not not work like that of max voting method.

Model	Model Parameters	Gini Score
Logistic Regression	random_state = 2020, penalty='l2'	0.21561
Random Forest	max_depth=6, min_samples_leaf=50	0.26178
Decision Trees	max_depth=6,min_samples_split=70	0.21730
XGBoost	n_estimators=1200, max_depth=6	0.29023

IV. Conclusion

With the Gini score of 0.29023, XGBoost model (without SMOTE) best predicted whether the insurance was claimed by the driver or not.

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear/_model.LogisticRegression.html
[2] https://scikit-learn.org/stable/modules/tree.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[4] https://builtin.com/data-science/random-forest-algorithm
[5] https://xgboost.readthedocs.io/en/latest/python/python_api.html
[6] https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/
[7] https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/