PS: Safe Driver Prediction

Team Name: .CSV

Nazar Kashif
IMT2018042
International Institute of Information Technology, Bangalore
mohdnazar.kashif@iiitb.org

Vinayak Agarwal
IMT2018086
International Institute of Information Technology, Bangalore
vinayak.agarwal@iiitb.org

Saad Patel
IMT2018514
International Institute of Information Technology, Bangalore
mohammad.saad@iiitb.org

***Abstract*—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. 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.**

This document is a detailed report on our work on predicting whether a driver will initiate an auto insurance claim in the next year, and also make auto insurance coverage more accessible to more drivers.

***Index Terms*—Data Preprocessing, OneHot Encoding, Label Encoding, Cross Validation, Logistic Regression, Random Forest Classifier, XGBoost Classifier, LightGBM Classifier, Randomized Search, Grid Search, EasyEnsemble Classifier, Stacking, Weighted Average**

INTRODUCTION

Porto Seguro, one of Brazil's largest auto and homeowner insurance companies, completely agrees with the idea that some drivers have to pay so much even if they have been 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.

In this competition, we were challenged to build a model that predicts the probability that a driver will initiate an auto insurance claim in the upcoming next year. A more accurate prediction will allow them to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

We will visualise the different data features, their relation to the target variable, explore multi-parameter interactions. The aim of this challenge is to predict the probability whether a driver will make an insurance claim, with the purpose of providing a fairer insurance cost on the basis of individual driving habits. It is an In-Class Competition organized by our professors and TAs to give us a hands on experience in real life data science.

I. DATASET

The dataset used in the project can be downloaded from here, which was hosted on the Kaggle platform.

The data consists of 57 input features, one target variable and a client ID. The target value is 1 if the client has filed a claim and it is 0 if not. Each row is a client and they are independent. 1,78,564 unique clients are present in test set while 4,16,648 are present in the test set.

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).

"Ind" is related to individual or driver, "reg" is related to region, "car" is related to car itself and "calc" is an calculated feature. In addition, feature names include the postfix 'bin' to indicate binary features and 'cat' to indicate categorical features and 'calc' to denote calculated features. Features without these designations are either continuous or ordinal. Values of -1 indicate that the feature was missing from the observation. 17 of the 57 features are binary, 14 are categorical the others are either continuous or ordinal.

There are no NaNs in either test or train data. But values of -1 indicate that the feature was missing

from the observation.

Given the categories and data it's understood that the problem falls under binary classification.

So here drivers can be classified into two categories as a) 0 and b) 1. Here class 1 represents a driver or client which initiates an auto insurance claim. Whereas class 0 represents a driver which will not initiate an auto insurance claim.

II. Observations

1) The target is highly imbalanced in the ratio 0.03791 (minority:majority) (Below Figure 1)

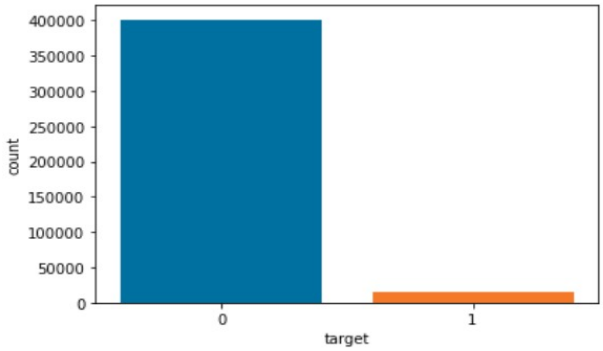

Fig. 1. Distribution of data over the target in the dataset.

2) Null values percentage : Many columns are found to have high missing values percentage. (Below Figure 2)
   a. ps_car_03_cat has around 69% missing values.
   b. ps_car_05_cat has around 44% missing values.
   c. ps_reg_03 has around 18% missing values.

	nullvaluesPercentage
ps_car_03_cat	69.089837
ps_car_05_cat	44.782531
ps_reg_03	18.106490
ps_car_14	7.160474
ps_car_07_cat	1.930237

Fig. 2. Null value percentages for features in descending order

3) We observed that all features are int or float type. Some features namely, ps_reg_03, ps_car_12, ps_car_13 and ps_car _14 have high number of distinct float values(Below Figure 3).

```
ps_reg_03 has 5013 distinct values
ps_car_12 has 184 distinct values
ps_car_13 has 70482 distinct values
ps_car_14 has 850 distinct values
```

Fig. 3. Count of unique values for some features

4) Some features have high correlation with other features, and some have low correlation with target variables, and we'll be dropping these during preprocessing. (Below Figure 4)

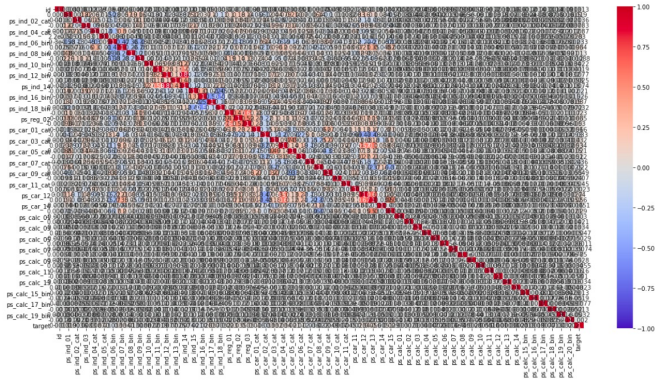

Fig. 4. Heatmap of correlation between target and features.

5) We observed that only some features like ps_reg_03, ps_car_12, ps_car_13, ps_car_14 have continuous data distribution. Rest other features have discrete data distribution. (Below figures 5 and 6)

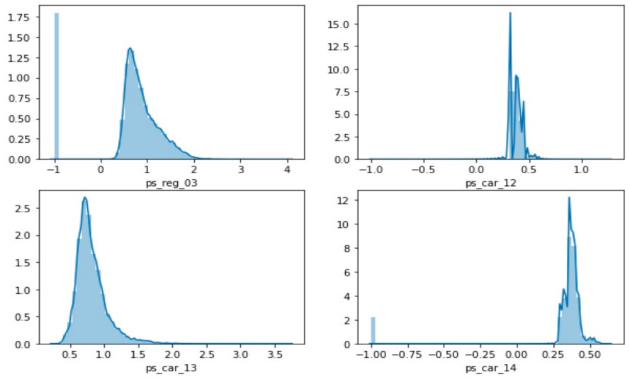

Fig. 5. Distribution plot of some continuous features

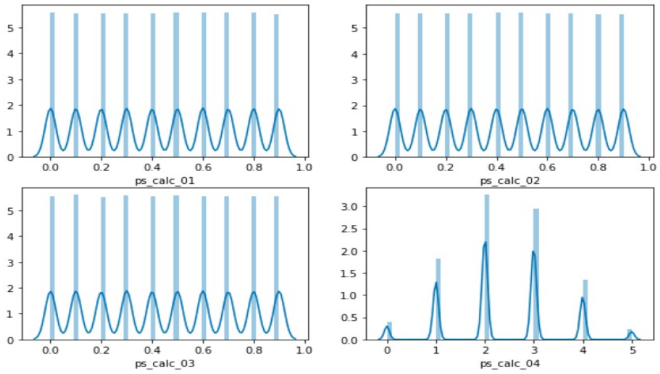

Fig. 6. Distribution plot of some calc features

6) Some features have a high imbalance ratio(majority:minority) as >= 0.95, namely ps_ind_10_bin, ps_ind_11_bin, ps_ind_12_bin and ps_ind_13_bin. (Below Figure 7)

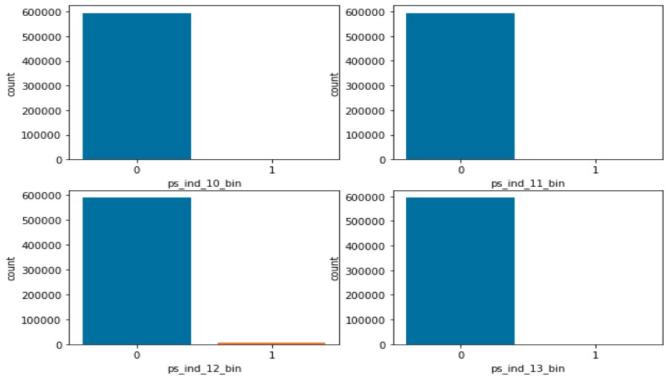

Fig. 7. Count plot indicating imbalance in feature values

7) Some features were found to have outliers namely, ps_reg_02, ps_reg_03, ps_car_11, ps_car_12, ps_car_13 and ps_car_14. (Below Figure)

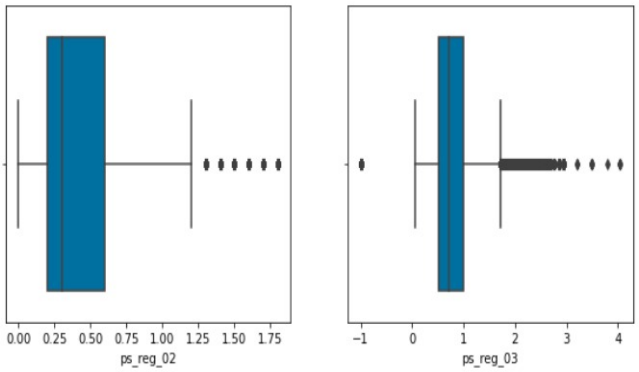

Fig. 8. Count plot indicating imbalance in feature values

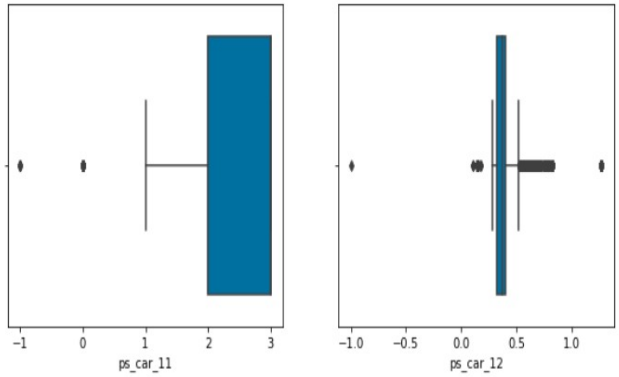

Fig. 9. Count plot indicating imbalance in feature values

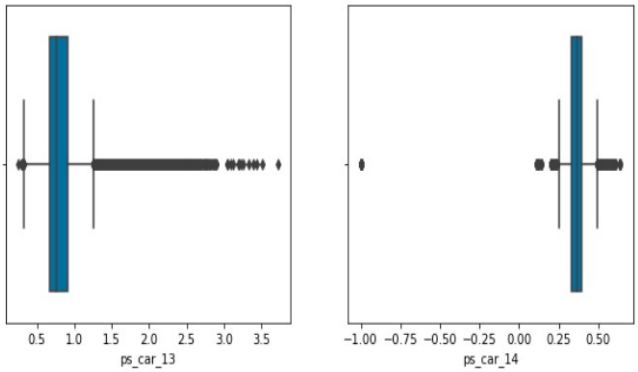

Fig. 10. Count plot indicating imbalance in feature values

III. Data Preprocessing and Feature Extraction

1) We combined train and test data to form a new dataframe and performed Data Preprocessing and Feature extraction on the combined dataframe to avoid inconsistencies during training and testing.

2) We dropped all calc features, because they are calculated features from other features, so they don't signify anything from their own.

3) We separated the data set into categorical, binary and other features for further analysing them with different techniques.

Binary data Preprocesing:

4) We dropped the features that have a high imbalance ratio >= 0.95, namely, ps_ind_10_bin and ps_ind_13_bin.
5) There were no null values, so we didn't do anything there.

Categorical data Preprocesing:

6) We filled the missing values in some categorical features like ps_car_03_cat and ps_car_05_cat which had about 50-70% missing values with a new category as we found that dropping them caused the gini score to drop a lot. We filled the missing values in remaining categorical features with the mode.

7) We applied One-hot encoding to the categorical features. We tried label encoding and a combination of both also, but it didn't perform as good as One-Hot encoding.

Continuous data Preprocesing:

8) We filled the missing values of all continuous features with mean.

9) We wrote a function to detect and remove outliers (remove_outliers()), but the gini score dropped drastically so we didn't do it for out final model. This may have happened because a lot of features had imbalance and the function was recognising the values with less count as outliers.

10) We checked for skew and tried to normalise data and remove skew by using log and power transforms, but the validation score decreased. We think this was happening because we were basically changing the meaning of the data. It was meant to be skewed.

11) Since initially we joined the train and test for preprocessing, we split it again into the submission test data, and train data.

Handling imbalance in train data

12) We tried oversampling the minority class, by using the resample from sklearn,but it performed miserably. This might be because we were just adding same datapoints to the minority class, leading to overfitting.

13) Then we tried undersampling the majority class, it didn't perform as bad as oversampling, but still performance was less than what we got without doing any kind of sampling. We think this might be because some important data points could be getting erased.

14) We also tried synthetic sampling using SMOTE and NEARFIT, but they were worse than the normal resampling technique.

15) After that we split the train data into cross validation sets. We trained a LightGBM model on the data and used it to calculate feature importance and cross validation gini score, then removed the features with 0 importance. This helped in reducing dimension of the data.

16) Finally, we trained the model on the whole dataset and got the predictions on the test data.

IV. Training Details and Results

We tried a variety of combinations of hyperparameters and then found the best combination using grid search and randomised search. We used cv = 5 and cv = kstatified fold(sklearn)

Since there were only 5 submissions per day, we had to take a lot of care that we don't overfit or underfit the data, so many times we tuned the models manually.

We tried stacking the combination of above Xgboost classifier and Lightgbm classifier but we got a low score. So we used imblearn library in which we found easy ensemble classifier to be the best. For the final model, we used both the grid search and randomized search for tuning hyperparameters.

Table I
Models Evaluated With Different Feature Sets

Model	Model Parameters	Normalized Gini Index
Logistic Regression	max_iter=1000, penalty=l2, random_state=0	0.2428
Perceptron(SGDClassifier)	loss="perceptron" learning_rate=0.001 penalty="l1"	0.179
Support Vector Machines	loss="hinge" learning_rate=0.001 penalty="l1"	0.167
Random Forest Classifier	n_estimators=1600, class_weight="balanced", min_samples_leaf=1000, max_leaf_nodes=150 n_jobs=-1	0.253
XGBClassifier	n_thread=-1, learning_rate=0.01, n_estimators=150, colsample_bytree=0.04, scale_pos_weight=3, eval_metric="auc", max_depth=20	0.264
LightGBMClassifier	objective='binary' n_estimators=1600 boosting_type='goss n_jobs=-1 col_bysample=0.04 learning_rate=0.005 max_bin=10	0.2753
EasyEnsemble Classifier+ XGBoost	n_estimators=45 base_estimator=xgboost random_state=42 n_jobs=-1 sampling_strategy='majority'	0.272
EasyEnsembleClassifier+light gbm	n_estimators=45 base_estimator=lightgbm random_state=42 n_jobs=-1 sampling_strategy='majority'	0.277

V. Models

A. Logistic Regression :

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression is estimating the parameters of a logistic model (a form of binary regression).

optimization problem hyper-parameter tuning

B. SVM:

It uses classification algorithms for two-group classification problems

C. Random Forest Classifier :

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

D. XGBoost:

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. It can be used to solve regression, classification, ranking, and user-defined prediction problems. It runs smoothly on all major OS and supports major programming languages.

E. LightGBM:

Light GBM is a gradient boosting framework that uses tree based learning algorithm. Light GBM is sensitive to overfitting and can easily overfit small data. (So preferable to large datasets)

VI. Conclusion

The best model that we got after training was EasyEnsembleClassifier using base_estimator as LightGBMClassifier.

The model predicts the probability of a driver claiming auto insurance in the next year with a Normalized Gini Index score of 0.274 on the private leader board. The visualizations gave an insight on how the data is distributed and behaves, and the predictions guided us towards improving the model.

Such projects have a potential scope to personalize the price of auto insurance for each driver depending on the prediction of target which will help car insurance company to raise the cost of insurance for bad drivers and reduce the price for good ones. This will help in fairness of insurance claims, and will ensure justice for the company as well as for the drivers.

VII. Challenges Faced

1) One of the major challenges was to deal with the imbalanced data, for which we followed various techniques as described in data preprocessing. But even those methods weren't useful, and that led us to explore more ways to handle imbalanced prediction. One way that we found and used, was the imblearn library and its EasyEnsembleClassifier.

2) Initially we were preprocessing the train and test data separately, so there were inconsistencies in the encodings of the categorical data in the train and test data. So, after that we tried merging the trained and test data set and preprocessed the merged data set.

3) There was a phase in the middle of the contest, during which we tried various techniques, but the score was stuck in a range. Then we used GridSearchCV, some manual hyperparamter training and some other ensemble techniques like weighted average, stacking etc. , which really improved the performance of the models.

VIII. Acknowledgement

Only the theory behind a concept can't help us gain insight on how it takes effect in real life. At the same time blindly implementing a thing without studying it first will never help us in the long run.

Projects are a bridge between theoretical and practical work. They help us understand how theory works in reality. We can't stress enough on the fact that how much this project helped us to understand the core concepts, and this was only possible because of our respected teachers, Professor G Srinivasaraghavan and Professor Neelam Sinha, and all our TAs who helped us throughout the course and the project. We want to give a special thanks to our project TA, Vibhav Agarwal, for giving us tips and clearing all our doubts that we had during the project. He was also understanding of our obligations and the organisation of the review meetings was flawless.

We would also like to thank our friends and colleagues with whom we had healthy discussions and arguments which helped us gain knowledge and grow , not only as data scientists , but also as people.

Finally we would like to thank Kaggle , for providing such a great platform to learn from and for making it so easy to implement things for the project. The Kaggle Community was instrumental in clearing some important doubts and making us feel a part of the Kaggle family. We would definitely keep contributing to it, and help fellow kagglers, to grow together and help the advancement of machine learning in the world.

IX. References

A. Books and Resources:

- DATA https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data
- MISSING VALUES https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779
- ONEHOT ENCODING www.machinelearningmastery.com sklearn.preprocessing.OneHotEncoder.html
- EVALUATION METRICS https://www.researchgate.net/publication/275224157
- XGBOOST https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python
- SVM VS LOGISTIC REGRESSION https://doi.org/10.1007/s13042-012-0068-x
- FEATURE SCALING https://sebastianraschka.com/Articles/2014_about_feature_scaling.html
- RESEARCH PAPER on the Features of Car Insurance Data Based on Machine Learning, by Hui Dong Wang from School of Information Engineering and Automation, Kunming University of Sciene and Technology, Kunming 650500, China https://www.sciencedirect.com/science/article/pii/S1877050920301381
- PAPER Chen, T. Guestrin, C., 2016. Xgboost: A scalable tree boosting system. arXiv preprint arXiv:1603.02754.

B. Tools and Libraries:

- SEABORN APPS https://seaborn.pydata.org/
- LABELENCODER https://scikit−learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
- ONEHOTENCODER https://scikit−learn.org/stable/modules/ generated / sklearn. preprocessing. OneHotEncoder. html
- XGBOOST https://xgboost.readthedocs.io/
- RANDOMFOREST CLASSIFIER https://scikit−learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- STANDARD SCALER https://scikit−learn.org/stable/modules/ generated/sklearn.preprocessing.StandardScaler.html
- RANDOMIZED SEARCH CV https://scikit−learn.org/stable/modules/generated/sklearn.model_selection. Randomized SearchCV.html
- GRID SEARCH www.scikit−learn.org/stable/modules/generated/sklearn.model_selection. GridSearch CV.html
- EASY ENSEMBEL https://imbalanced−learn.org/stable/generated/ imblearn.ensemble.EasyEnsemble Classifier.html