Porto Seguro: Safe Driver Prediction

Team : Matrix

Urja Srivastava
MT2020037
IIIT Bangalore
urja.srivastava@iiitb.org

Mehak Dogra
MT2020090
IIIT Bangalore
mehak.dogra@iiitb.org

Shubhi Maheshwari
MT2020167
IIIT Bangalore
shubhi.maheshwari@iiitb.org

***Abstract*—The problem given in the competition is a Machine Learning problem to predict the probability whether a driver will initiate an auto insurance claim in the next year. This is checked on the basis of different properties/characteristics of the machine.**

***Index Terms*—Exploratory data analysis, Data preprocessing, Feature engineering, encoding, Model Ensembling, Stacking.**

Problem Statement

Nothing can ruin the thrill of buying a new car more quickly than seeing a new insurance bill. The sting is more painful when you know you drive carefully and are a good driver. It does not seem fair that you have to pay so much if you are a good driver.

Porto Seguro, one of Brazil's largest auto and homeowner insurance companies, totally agrees. Insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for average or bad ones.

In this competition, we are challenged to build a model that predicts the probability that a driver will initiate an auto insurance claim in the next year. While Porto Seguro has used machine learning but we were supposed to explore new and more powerful methods. A more accurate prediction will allow them to further tailor their prices, and make auto insurance coverage more accessible.

Dataset

(416648, 59) The data set given in the competition contain multiple feature properties and each row in this dataset corresponds to a different feature, uniquely identified by a column named ID. There are 59 columns in the dataset. On studying this dataset we came to know that the null values in this dataset are represented with an integer i.e. -1.

While the dataset provided here has been roughly split by time, the complications and sampling requirements mentioned above may mean you may see imperfect agreement between your cross validation, public, and private scores!

Uniquely identified ID in each row corresponds to Target indicating that Insurance was claimed by a driver or not. Using the information and labels in train.csv, you must predict the value for Target for data provided in test.csv.

I. Introduction

For the purposes of safety and security, Law has made it mandatory that all the people owning a vehicles should possess Car insurance. This is because when a person has the insurance he can protect himself from the many risks related to finance and expense that he has to bear when an accident takes place.

The matter of concern is not all people need to pay the bulky amount of the insurance when they know that they are a good and careful drivers. The question of affordability arises when a careful good driver has to face the unexpected expenses of this insurance bills.

But using certain features it can be predicted whether a person will file a claim for the insurance in the next year. A more accurate prediction will allow them to further tailor their prices, and make auto insurance coverage more accessible.

We are given with such features using which our model can predict the same. We are supposed to build a model which gives a better result in order to get make insurance coverage more accessible.

II. Data Visualization

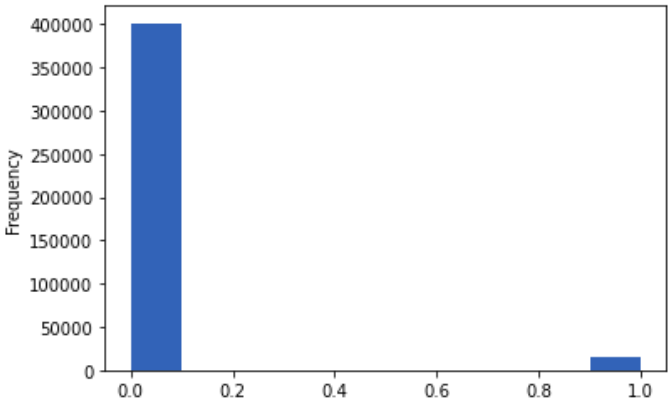

Fig. 1. Target Analysis

The dataset consists of features as columns. These are different kinds of data types that is int64 or float64 while other are categorical data. In the dataset, there are certain features that have very high number of missing values(more

than 60 percent).There is a high amount of correlation (higher than 0.7) amongst the numerical features. In the object data types, there is large number of categories.

- Features that belong to similar groupings are tagged as such in the feature names (e.g., ind, reg, car, calc).
- 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.

A. Individual Feature Visualisations

- Binary features ps_ind_06_bin, ps_ind_07_bin, ps_ind_08_bin, ps_ind_09_bin, ps_ind_10_bin, ps_ind_11_bin, ps_ind_12_bin, ps_ind_13_bin, ps_ind_16_bin,ps_ind_17_bin, ps_ind_18_bin,ps_calc_15_bin, ps_calc_16_bin, ps_calc_17_bin, ps_calc_18_bin, ps_calc_19_bin, ps_calc_20_bin

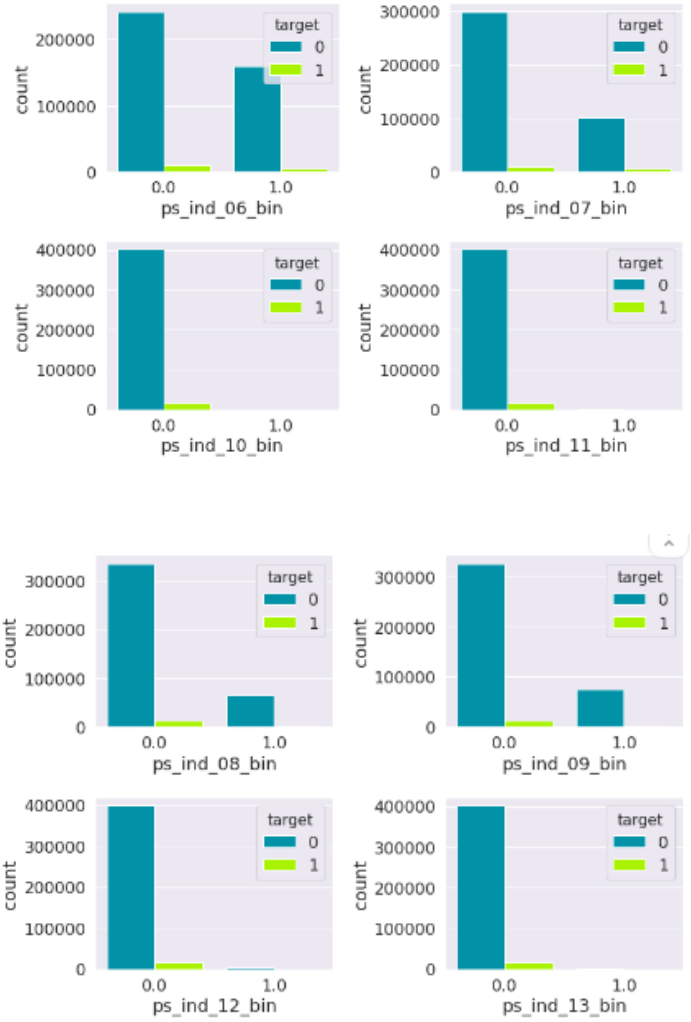

Fig. 2. Binary Features

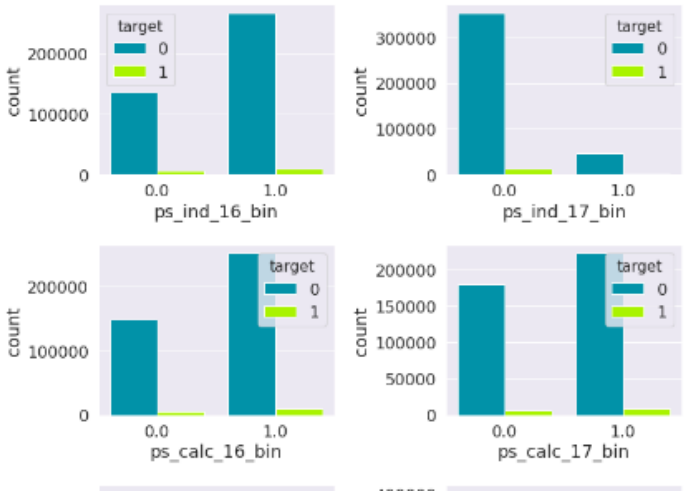

Fig. 3. Binary Features

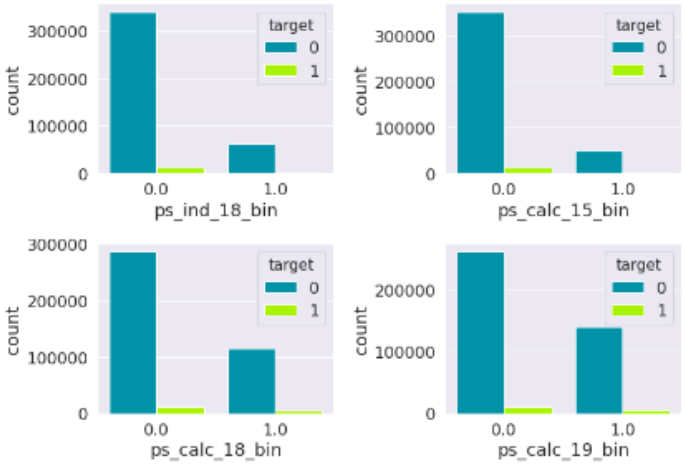

Fig. 4. Binary Features

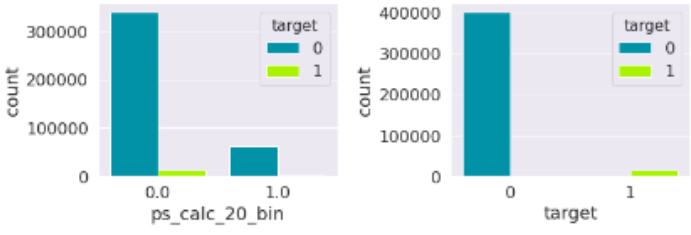

Fig. 5. Binary Features

- categorical features

  ps_ind_02_cat, ps_ind_04_cat, ps_ind_05_cat,
  ps_car_01_cat, ps_car_02_cat, ps_car_04_cat,
  ps_car_05_cat, ps_car_06_cat, ps_car_07_cat,
  ps_car_08_cat, ps_car_09_cat, ps_car_10_cat,
  ps_car_11_cat

  Let's see the count plots of these categorical features...

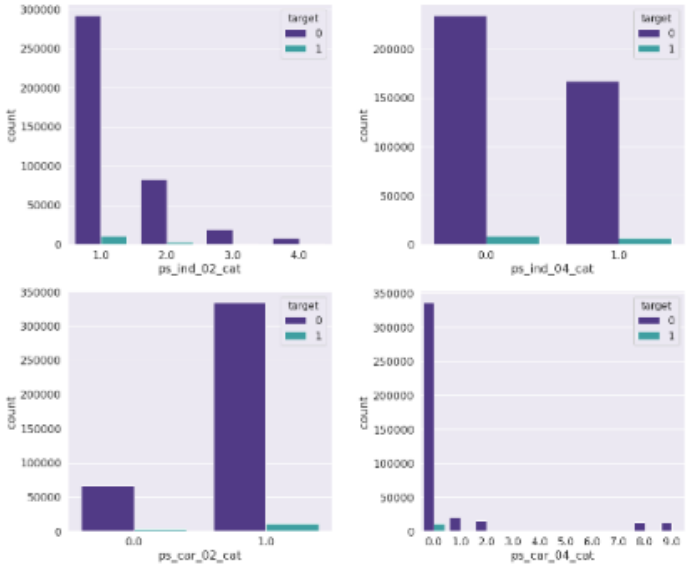

Fig. 6. Categorical Features

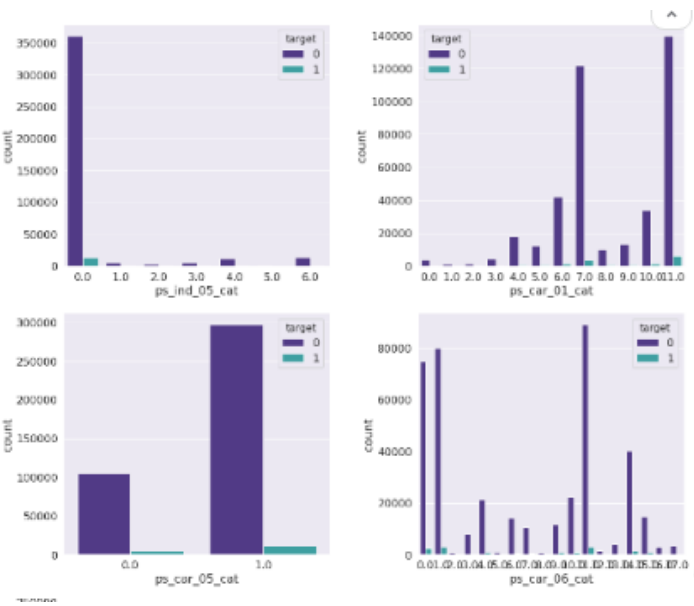

Fig. 7. Categorical Features

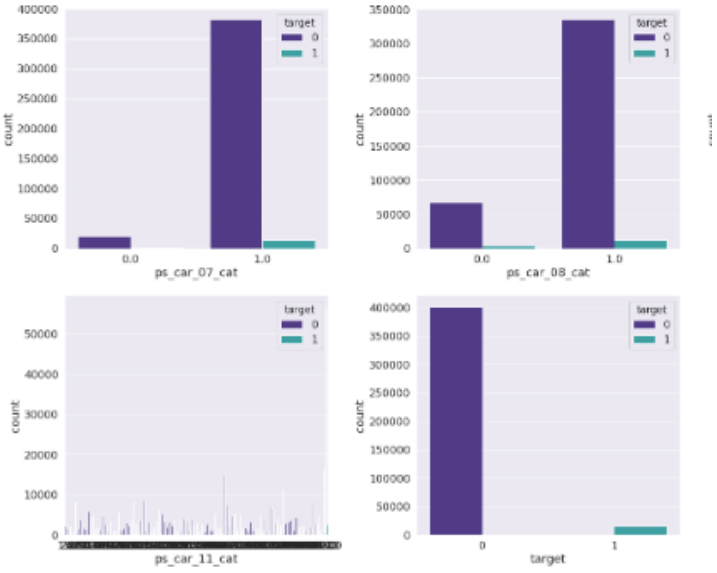

Fig. 8. Categorical Features

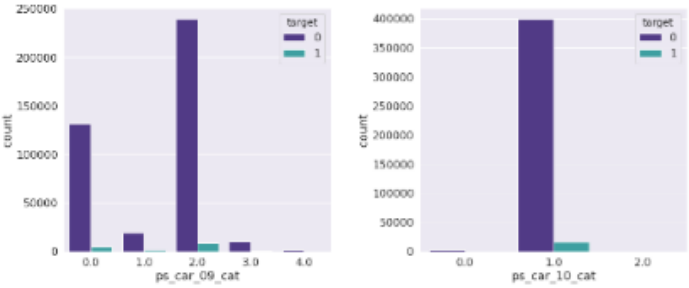

Fig. 9. Categorical Features

- Normal features

  ps_ind_01, ps_ind_03, ps_ind_14, ps_ind_15, ps_reg_01, ps_reg_02, ps_reg_03, ps_car_11, ps_car_12, ps_car_13, ps_car_14, ps_car_15, ps_calc_13

  Lets see the plots of these normal features...

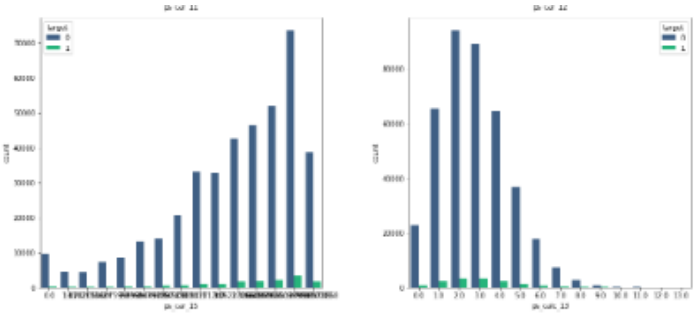

Fig. 10. Normal Features

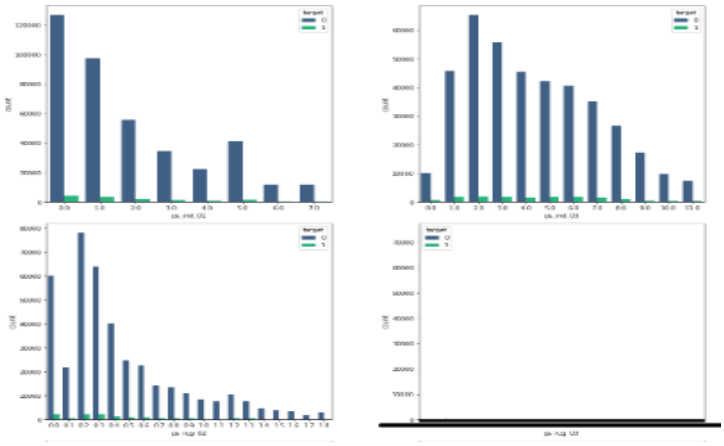

Fig. 11. Normal Features

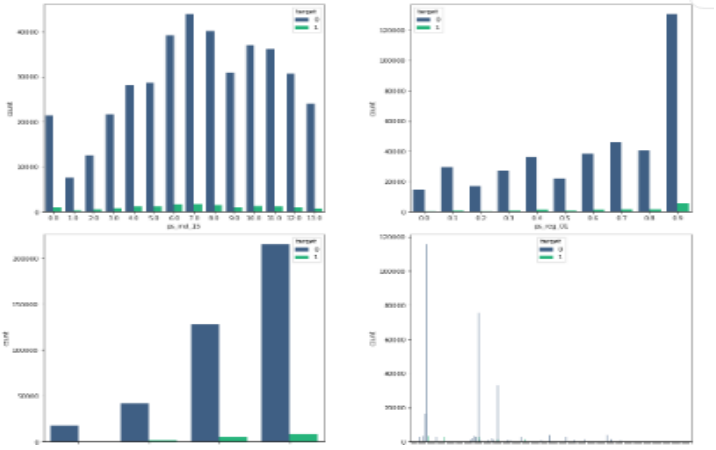

Fig. 12. Normal Features

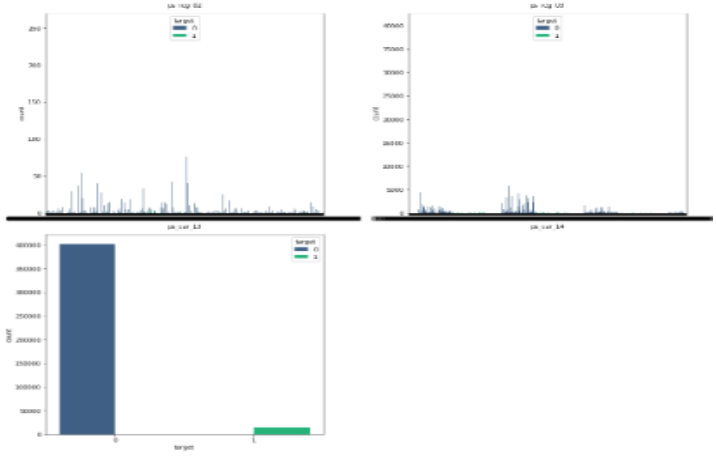

Fig. 13. Normal Features

III. Data cleaning and Pre-processing

There are total of 59 features in the train dataset while in test dataset there are 58 features. It is observed that Target feature is missing. It is also observed from the dataset that the data is highly skewed and contains a lots of missing values. In the dataset, there are both numerical as well as categorical features.

Handling Missing Data

Before starting with this step we need to change all the representation of the null values. As mentioned above the missing values are represented by -1 value in the dataset. We will change this to the representation saying ”NA” using ”dataset.replace(-1,np.nan)”. This is a necessary step to check for the missing values using ”dataset.isnull().sum()”.

Then we will check for the percentage of the missing values in the dataset by isnull() option from where we got to the percentage of NAN in each column. Later, we selected those rows and handled them accordingly.

```
ps_ind_02_cat has 148 records (0.036%)
ps_ind_04_cat has 55 records (0.013%)
ps_ind_05_cat has 4047 records (0.971%)
ps_reg_03 has 75476 records (18.115%)
ps_car_01_cat has 76 records (0.018%)
ps_car_02_cat has 2 records (0.000%)
ps_car_03_cat has 288186 records (69.168%)
ps_car_05_cat has 186614 records (44.789%)
ps_car_07_cat has 8095 records (1.943%)
ps_car_09_cat has 389 records (0.093%)
ps_car_11 has 1 records (0.000%)
ps_car_14 has 29822 records (7.158%)
There are 14 variables with missing values
```

Fig. 14. Missing Values %

A. *Dropping*

```
Highest missing values in ps_car_03_cat
Total missing values in ps_car_03_cat is 288186
percentage of missing values is (69.168%)
```

Fig. 15. Columns with highest missing values %

For the columns with more than 50percent of the missing values we dropped the column. There is only one column with over 50percent of NAN values. In the figure below we can see the columns which were dropped in the process. The name of that column which we dropped is ”ps_car_03_cat”.

Along with this we stored Target column into other new table/ array and ID too. Then we dropped these two columns from the datset to study other features properly.

Also we tried dropping the column which is highly correlated to the target. This is because we cannot get

anything new from the highly correlated features. And it is always better to drop the features which do not add something new to our learning. The column which we drop due to high correlation is ”ps_ind_14”.

B. Imputation

To handle missing data amongst the numeric features we employ mode/ most_frequent strategy of imputation. For categorical features Label Encoder automatically assigns a numeric category to missing (NaN) values.

IV. Feature Engineering

Feature Engineering and handling of features by data type. We divided the dataset in smaller datasets depending on their feature types. Firstly We worked with categorical data which had the datatype ”object”.On making this division We got 26 separate columns. On those columns we applied different techniques to extract more information from them. We first split the categorical data and stepped further with encoding. Similarly we did get a separate dataset for int data type and other for float data type.

A. Principal Component Analysis

On our dataset we even tried the technique called PCA which is used to decrease the dimensionality of the dataset. This technique is used many times for dimensionality reduction in machine learning.

High dimensionality means that the dataset has a large number of features. High-dimensionality in the machine learning field means the model is overfitting, which reduces the ability to generalize beyond the examples in the training set.

The ability to predict correctly exponentially decreases as the dimensionality of the training dataset increases. Models also become more efficient as the reduced feature boosts the learning rates.

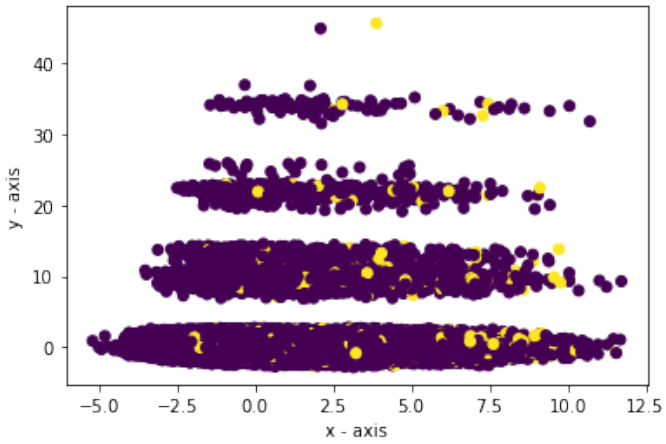

Fig. 16. Before PCA was applied

PCA is generally used to filter noisy datasets, such as image compression. Each additional attribute expresses less variance and more noise, so representing the data with a smaller subset of principal components preserves the information and discards the noise.

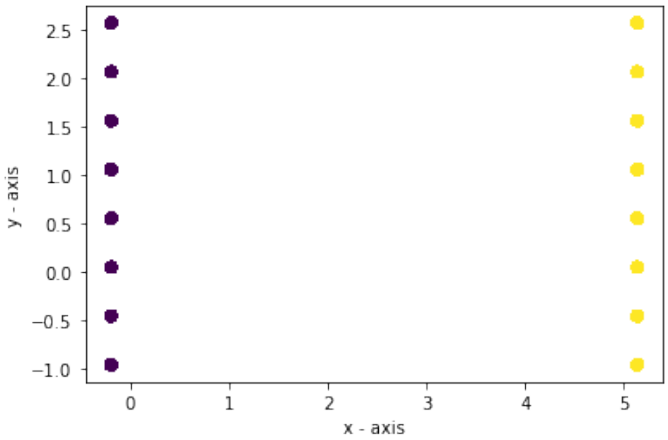

Fig. 17. After PCA with component 50 was applied

We didn’t get some really good result result with PCA thus we changed the approach.

B. Feature Selection

In other notebook,for the numerical data, we observed a very high correlation. To dropped the columns that we need to remove which were highly correlated i.e., columns with correlation values greater than 0.8 in order to avoid multidimensionality in the data-set.

We divided the data into three main parts i.e. binary categorical and normal dataset. These datasets contained the respective features of the main dataset which were binary features, categorical features and normal features.

We also used different techniques of feature selection.

- Variance Threshold
- Chi-square method
- Forward Feature Selection
- Backward Feature Elimination
- Recursive Feature Elimination

C. Variance Threshold

The variance threshold is a simple baseline approach to feature selection. It removes all those features whose variance doesn’t meet the provided threshold. By default, it removes the features that have the same value in the dataset. This method takes into consideration that features with a higher variance may contain more useful information, but note that we are not taking the relationship between feature and target variables into account, which is one of the drawbacks of this method.

```
selected=pd.Series(portobin)[thresholder.get_support()]
selected
```

```
0      ps_ind_06_bin
1      ps_ind_07_bin
8      ps_ind_16_bin
12    ps_calc_16_bin
13    ps_calc_17_bin
14    ps_calc_18_bin
15    ps_calc_19_bin
dtype: object
```

Fig. 18. Resultant of Variance Threshold on Binary dataset

D. Chi-square method

The Chi-square test is used on categorical features. We calculate Chi-square between each feature and the target using "chi2_selector = SelectKBest(chi2, k=6)" and select the features with the best Chi-square result. In order to apply the chi-squared the following conditions have to be met:

- The variables have to be categorical
- Sampled independently
- Values should have an expected frequency greater than 5

```
selectedcat=pd.Series(portocat)[chi2_selector.get_support()]
selectedcat
```

```
2     ps_ind_05_cat
3     ps_car_01_cat
4     ps_car_02_cat
5     ps_car_04_cat
6     ps_car_06_cat
11    ps_car_11_cat
dtype: object
```

Fig. 19. Resultant of Chi-Square on Categorical dataset

E. Forward Feature Selection

This is an iterative method where we select a feature, if gives good result, we start with that against the target. Then, we select another feature that gives the best performance in combination with the earlier selected feature. This process continues until the preset criterion is achieved. In this process each feature is selected manually. This is a time consuming process and many times we don't end up with a good result.

F. Backward Feature Elimination

This method works exactly opposite to the Forward Feature Selection method. Here, we begin with all the features available and build a model and check the score. Then, we remove a feature from the model and again train and test it. If the result is improved we continue else we add that feature again and remove another and repeat the process.

G. Recursive Feature Elimination

RFE is popular because it is easy to configure and use and because it is effective at selecting those features (columns) in a training dataset that are more or most relevant in predicting the target variable. There are two important configuration options when using RFE: the choice in the number of features to select and the choice of the algorithm used to help choose features. Both of these hyperparameters can be explored, although the performance of the method is not strongly dependent on these hyperparameters being configured well.

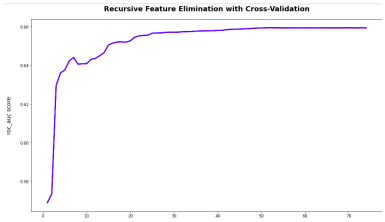

Fig. 20. Recursive Feature Elimination

Then we checked the correlation in each dataset.

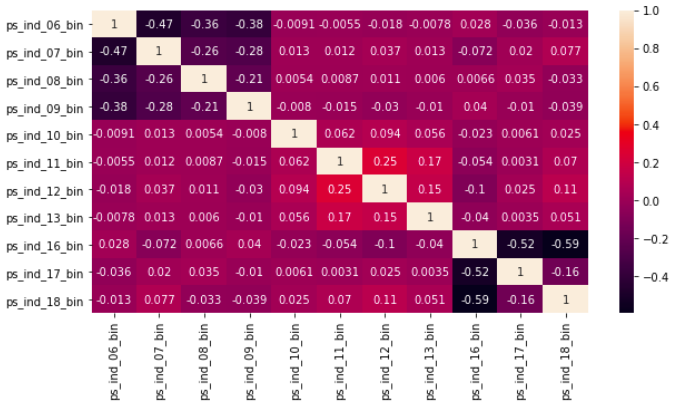

Fig. 21. HeatMap-Correlation Matrix of Binary dataset

Fig. 22. HeatMap-Correlation Matrix of Categorical dataset

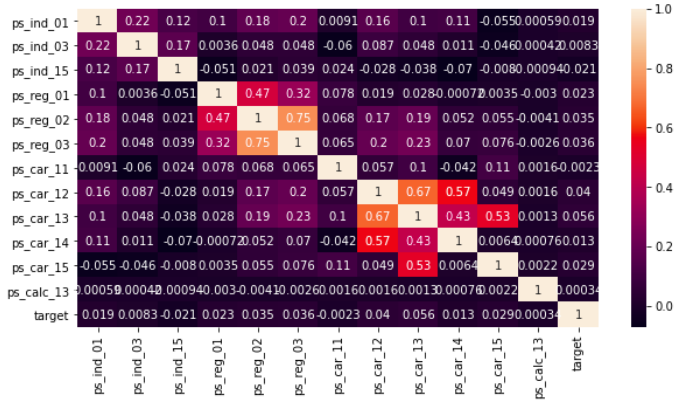

Fig. 23. HeatMap-Correlation Matrix of normal dataset

H. Encoding

We applied three type encoding techniques.

- One Hot Encoding
- Frequency Encoding

The given dataset was already label encoded. On the categorical data, after the split, we used Hot Encoding technique to convert the categorical data into a form that could be provided to ML algorithms to do a better job in prediction. After that we used Frequency Encoding in which machine replaces the category by the frequency or percentage of observations in the dataset. In the end we concatenated the spilt dataset and the dataset we get after encoding the categorical variables. Also used target encoding for top categorical features we got from the feature importance graph of lightgbm model.

I. Merging Different DataSets

After all divisions of subsets and editing of the datasets of different data types we merged all the data sets to form a single data set on which the final model will work.

While dividing the data we dropped calc features because they didnt give some good result. Rather we got better result without these features. So we dropped them in order to improve our model prediction.

J. Adding new Features

Extensive feature engineering. Took the top 5 features from lightgbm model feature importance graph and combined them in different ways. Added interaction features between top features. Grouped categorical variables and generated aggregate features by taking mean,median etc. Generated two features -ind,car, which proved to be very good for improving gini score.

V. Model Ensembling

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction.

A. Naive Bayes

it assumes that the occurrence of a particular feature is independent of the occurrence of other features. Such as if the machine is identified on the bases of model, speed, shape and accuracy, then , latest, fast, compact, and quite accurate machine is recognized as a mobile. Hence each feature individually contributes to identify that it is a mobile without depending on each other.

B. Logistic Regression

The logistic classification model (or logit model) is a binary classification model in which the conditional probability of one of the two possible realizations of the output variable is assumed to be equal to a linear combination of the input variables, transformed by the logistic function.

C. Bagging classifier

A Bagging classifier is an meta-estimator that fits base classifiers on each random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Each base classifier is trained in parallel with a training set. Many of the original data may be repeated in the resulting training set while others may be left out.

D. Decision trees

Tree models where the target variable can take a discrete values are called classification trees. Decision trees where the target variable can take continuous values are called regression trees. Classification And Regression Tree is general term for this.Decision trees are constructed with an algorithm that identifies ways to split a data set based on different conditions. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks.

E. Light GBM

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.

F. XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve problems in a fast and accurate way.

G. Catboost

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy. 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.

H. Stratified K-Fold

Approach used for model evaluation is the train/test split and the k-fold cross-validation procedure. Both approaches can be very effective in general, although they can result in misleading results and potentially fail when used on classification problems with a severe class imbalance. Instead, the techniques must be modified to stratify the sampling by the class label, called stratified train-test split or stratified k-fold cross-validation.Each fold tries to depicts or duplicates the original data.

VI. Hyperparameter Tuning

Hyperparameters are important for machine learning algorithms as they control the behaviors of training models and have a significant impact on the performance of machine learning models.There are different techniques used for the tuning of hyperparameters. Some used techniques are.

A. Grid Search

This is a time consuming process as it tries all the possible configurations of the parameters. In the end it returns the configuration which provides the best score. It is best to use it if the dimensions are less.

B. Bayesian Optimization

Bayesian Optimization provides a principle technique based on Bayes Theorem to search a solution of a problem that is efficient and effective. It works by building a probabilistic model of the objective function, called the surrogate function, that is then searched efficiently with an acquisition function before the samples are chosen for evaluation on the real objective function.

VII. Training the model

After feature engineering, our number of useful features reduced.Reducing the features also decreased the complexity of the model as well.After modifying the model parameters we could improve the model accuracy.Also we tried different supervised learning models to improve our accuracy.We trained our data on the following algorithms:

- Linear Model(LM)
- Naive Bayes
- Support Vector Machine(SVM)
- Logistic Regressor(LR)
- Bagging classifier
- Decision trees
- LightGBM(LGBM)
- XGBoost(XGB)
- Stratified KFold(SKFold)

S.No.	Algorithm	gini score
1.	Naive Bayes	0.24248
2.	Logistic Regression(LR)	0.25502
3.	Bagging Classifier with LR	0.27689
4.	XGB	0.29365
5.	LGBM	0.29230
6.	CatBoost	0.28968
7.	Ensembling(average)	0.29399
8.	Random Forest	0.26888
9.	Stacking Classifier	0.29369
10.	Stratified KFold CatBoost	0.28926
11.	Voting Classifier	0.29312

The k value must be chosen carefully for your data sample.A poorly chosen value for k may result in a mis-representative idea of the skill of the model, such as a score with a high variance,or a high bias, (such as an overestimate of the skill of the model).

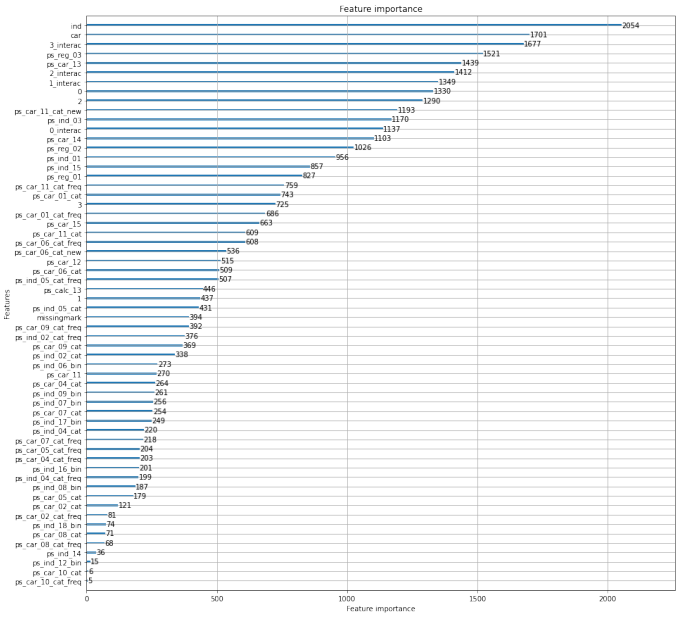

Fig. 24. Execution Importance

Our engineered features are doing very well on the feature importance graph and gave us that boost in prediction accuracy.

VIII. Conclusion

Using Light GBM,catboost and XGB ensembling, our model was able to predict the probability that a driver will initiate an auto insurance claim in the next year or not with

a gini score of .29399.Also we could predict which are the more dominant features that helps in better prediction thereby increasing the model accuracy. Ensembling gives more stable models which generalize better. Feature engineering boosts prediction.

	id	target
0	1225226	0.046918
1	487111	0.021282
2	1328243	0.080795
3	388051	0.022525
4	1454785	0.031416
...	...	...
178559	241164	0.019024
178560	720555	0.044272
178561	65816	0.016741
178562	538640	0.021580
178563	841373	0.023158

178564 rows × 2 columns

Fig. 25. Prediction of our final model

IX. Acknowledgement

We would like to thank Professor G. Srinivas Raghavan and our Machine Learning teaching assistant Vibhav Agarwal,for giving us the opportunity to work on the project and helping us out in the initial stages in numerous occasions. His highly detailed lectures on various topics helped us understand what we were actually doing.

Leader-board was great motivation to work on the project and the competition forced us to read up various articles and papers which gave us ideas and enthusiasm for the project.

X. References

[1] Jovan Sardinha. An introduction to model ensembling. [Online]. Available: https://medium.com/weightsandbiases/an-introduction-to-model-ensembling-63effc2ca4b3

[2] Philip Hyunsu Cho, Nan Zhu et.al., XGBoost. [Version: 1.2.0]. [Online]. Available: https://xgboost.readthedocs.io/

[3] Jérémie du Boisberranger, Joris Van den Bossche et.al., sckit-learn: Open source scientific tools for Machine Learning. [Latest] [Online]. Available: https://scikit-learn.org/

[4] Microsoft Corporation Revision 9597326e. LightGBM. [Latest]. [Online]. Available: https://lightgbm.readthedocs.io/

[5] Andrei (Andrey) Khropov, annaveronika et.al., catboost.ai. [version: 0.24.3]. [Online] https://github.com/catboost/catboost