A PROJECT REPORT ON

Driver Insurance Claim Prediction

A project report submitted in fulfilment for the Diploma Degree in AI & ML

Under

Applied Roots with University of Hyderabad

Project Submitted By

Surya Prakash Bhoi

Enrolment Number: 40AIML614-21/1

Under the Guidance of

Mentor: Aniket Vishnu

Approved by: Mentor: Aniket Vishnu

University of Hyderabad

Declaration of Authorship

We hereby declare that this thesis titled “Driver Insurance Claim Prediction” and the work is presented by the undersigned candidate, as part of Diploma Degree in AI & ML.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Surya Prakash Bhoi

Thesis Title: Driver Insurance Claim Prediction

CERTIFICATE OF RECOMMENDATION

We hereby recommend that the thesis entitled “Driver Insurance Claim Prediction” prepared under my supervision and guidance by Surya Prakash Bhoi, be accepted in fulfilment of the requirement for awarding the degree of Diploma in AI & ML Under Applied Roots with University of Hyderabad. The project, in our opinion, is worthy for its acceptance.

_____

Mentor: Aniket Vishnu

Under Applied Roots with

University of Hyderabad

ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of several wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success. I, Surya Prakash Bhoi, student of Applied Roots, am extremely grateful to mentors for the confidence bestowed in me and entrusting my project entitled “Driver Insurance Claim Prediction” with special reference.

At this juncture, I express my sincere thanks to Mentor: Aniket Vishnu of Applied Roots for making the resources available at right time and providing valuable insights leading to the successful completion of our project who even assisted me in completing the project.

Name: Surya Prakash Bhoi

Table of Contents

Abstract

Porto Seguro, one of Brazil's largest auto and homeowner insurance companies is currently facing the problem of inaccuracies in car insurance company's claim predictions, which results in increased cost of insurance for good drivers and reduces the price for bad ones.

This problem is equally shared by the insurer (the insurance company) and the insured (the insurance policy holder).

We would solve this business problem by predicting the probability of an insured filing an insurance claim next year. We will be using various Classification machine learning algorithms in approaching this problem, and select the best model among them by using Normalized Gini Coefficient as the metric.

Project Introduction

Business Problem

The problem that we are trying to solve is being faced by the insurers (automobile insurance companies) and the insured (the person who is covered against risk) equally. The problem is the “insurance premium amount” – where ideally, safe drivers should be charged a lower insurance premium for being more responsible drivers on road and are less viable to make accidents. And unsafe drivers should be charged higher insurance premium amount for being careless or insensitive towards the traffic rules, and are more prone to make accidents on the road.

Business wise, insurance companies face the risk of losing safe drivers as customers if they are charged with higher insurance premiums. And also face the risk of losing money through insurance claims made from unsafe drivers, if they were charged lower insurance premium amount.

To tackle this problem, we can come up with a solution which finds out whether a driver is likely to file a insurance claim next year, based on the driver’s available historical data. So that, the solution can be translated as:

- Low risk customers (i.e., drivers who are unlikely to file a claim next year) are charged lower premium amount.
- High risk customers (i.e., drivers who are likely to file a claim next year) are charged higher premium amount.

Challenges

The main challenge which was observed in this problem was its dataset, as the feature names as “masked” instead of real feature names. As a result of this, our analysis of the dataset will not be more interpretable. For e.g., we cannot conclude observable analysis results like, whether – commercial vehicle drivers are more likely to file claims than personal vehicle drivers, or whether male or female drivers file more insurance claims, or whether elder people are less likely to files insurance claim than younger people etc.

Approach to be followed in solving this problem

The solution to the problem is to build a machine learning model that would predict the probability that a driver would file an insurance claim next year. This being a classification task we will use the below classification algorithms.

- Logistic Regression
- K Nearest Neighbour
- Naïve Bayes
- Support Vector Machines
- Decision Trees

- Random Forest
- XG Boost Classifier

Once we are done with the models using above algorithms, we use the best among them by using the given evaluation metric.

Dataset

Source of dataset
The source of the dataset is the Kaggle problem "Porto Seguro's Safe Driver Prediction - Predict if a driver will file an insurance claim next year."

Dataset source URL:

https://www.kaggle.com/competitions/porto-seguro-safe-driver-prediction/data

Dataset properties
The dataset bundle has 3 files:

- train.csv – This is the dataset which we would be using for our entire model building. It has 59 columns (including the "id" and "target" variables), and 595212 observations. So, we have 57 columns to be used as input features. The target variable has values as 1 or 0 denoting whether the driver filed an insurance claim the following year. The size of this file is 115.85 MB
- test.csv – This is the dataset for which we need to do the prediction, and submit our solution. It has 58 columns (one 'id' column and remaining 57 as input features), and 892816 observations, for which we need to predict the "target" variable. The "target" here is to predict the probability of the driver filing an insurance claim next year. The size of this file is 172.01 MB
- sample_submission.csv – This is the sample file, specifying the format in which the predictions of test.csv needs to be submitted. This has only 2 columns – id and target. The "target" here is our predicted probability of the "id" driver filing an insurance claim next year.

In the train and test datasets, 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 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.

Dataset challenges
The challenge with the dataset is the feature names as "masked" instead of real feature names. As a result of this, our analysis of the dataset will not be more interpretable.

The train dataset is highly imbalanced with about 88% having target=0 (i.e., 88% of policy holders not filing an insurance claim in the following year).

There are three input features namely, 'ps_reg_03', 'ps_car_03_cat' and 'ps_car_05_cat' which have significant missing values. We will need to handle those missing values.

Evaluation metrics

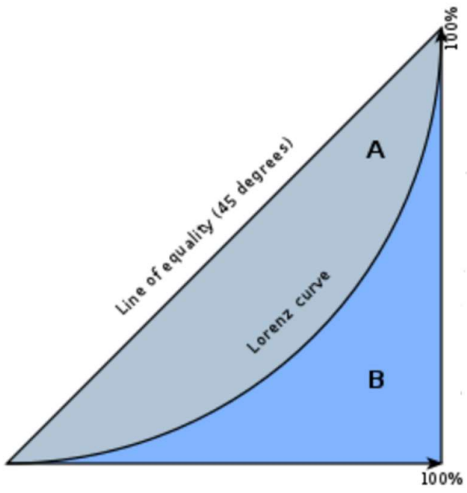
The scoring metric for our problem statement is "Normalized Gini Coefficient".

Gini Coefficient

Gini coefficient is a very popular metric in classification tasks, having imbalanced class values. It was developed by the statistician and sociologist Corrado Gini.

The Gini coefficient measures the inequality among values of a frequency distribution (e.g., levels of income within a nation or group). A Gini coefficient of 0 expresses perfect equality, where all values are the same (i.e., where everyone has the same income) while a Gini coefficient of 1 (or 100%) expresses maximal inequality among values (i.e., for a large number of people where only one person has all the income or consumption and all others have none, the Gini coefficient will be nearly one).

Sometimes a percentage representation of this coefficient is used, called the Gini index (the value varies from 0% to 100%).



Graphical representation of the Gini coefficient

The graph shows that the Gini coefficient is equal to the area marked A divided by the sum of the areas marked A and B, that is, Gini = A/(A + B). It is also equal to 2A and to 1 − 2B due to the fact that A + B = 0.5 (since the axes scale from 0 to 1).
Source: wikipedia

Calculation of Gini Coefficient

The formula for Gini Coefficient is:

$$G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2\sum_{i=1}^{n}\sum_{j=1}^{n}x_j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2n\sum_{j=1}^{n}x_j} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2n^2\bar{x}}$$

Where:
G = Gini coefficient
$\bar{x}$ = Average prediction
n = Total number of observations
x_i ***and*** y_i = The value of an individual's prediction

Disadvantage of Gini Coefficient
The problem with the Gini coefficient is that while it gives you a number to indicate how much inequality there is (0 = complete equality, 100 = very very unequal!), it won't say anything about the nature of the inequality.

Normalized Gini Coefficient
The Normalized Gini Coefficient adjusts the score by the theoretical maximum so that the maximum score is 1.

The formula for Normalized Gini Coefficient is:

$$Gini_{normalized} = \frac{Gini_{model}}{Gini_{perfect}} = \frac{(1-\pi)AUCROC + \frac{1}{2}}{\frac{1}{2}(1-\pi)} = 2AUCROC - 1$$

Where:
$Gini_{model}$ = Gini coefficent(actual values vs predicted values)
$Gini_{perfect}$ = Gini coeffcient(actual values vs actual values)
$Gini_{normalized}$ = Normalized Gini

The Normalized Gini Coefficient adjusts the score by the theoretical maximum so that the maximum score is 1

Real world constraints and expectations

Latency requirements
As the model is to be used as a system to predict the probability of an insured filing an insurance claim next year, the system will be used within an organization and will be used by associates or managers working inside the organization. So, it is ok if the model is able to generate the prediction within a few seconds or minutes.

Expectations

The model should be portable enough to be deployed as APIs, so that associates or managers of the organization can use it within browsers.

Exploratory Data Analysis and Feature Selection

Basic Information about our dataset

As part of basic information of our dataset, we found that:

- The dataset has 595212 observations, and 59 features. Out of the 59 features, one is the 'id' column and one id our 'target' variable. So, the effective number of input features in our dataset is 57.
- All the features are numeric (either int64 or float64)
- All feature names have a nomenclature, as below:
    - Features having 'bin' in their name are binary features – 17 features
    - Features having 'cat' in their name are categorical features – 14 features
    - Features without having 'bin' or 'cat' in their names are continuous features – 26 features
- The target variable 'target' is a binary feature with values either 0 or 1.
- Our dataset is highly imbalanced with less that 4% observation as positive (1), and more than 96% observations as negative (0).
- There are no duplicate observations in our dataset.
- The numeric features are not in any specific scale. So, we would need to scale them before modelling.
- 13 input features have missing data values. The missing values are already available as "-1" in the dataset.

Distributions of our data

From the distributions, i.e., histograms of the numeric features we observe that:

- The features - ps_calc_01, ps_calc_02 and ps_calc_03 follow an approximate uniform distribution.
- The features - ps_calc_07, ps_calc_10, ps_calc_11, ps_calc_14 follow an approximate normal distribution.
- And the remaining numeric features do not follow any specific distribution.

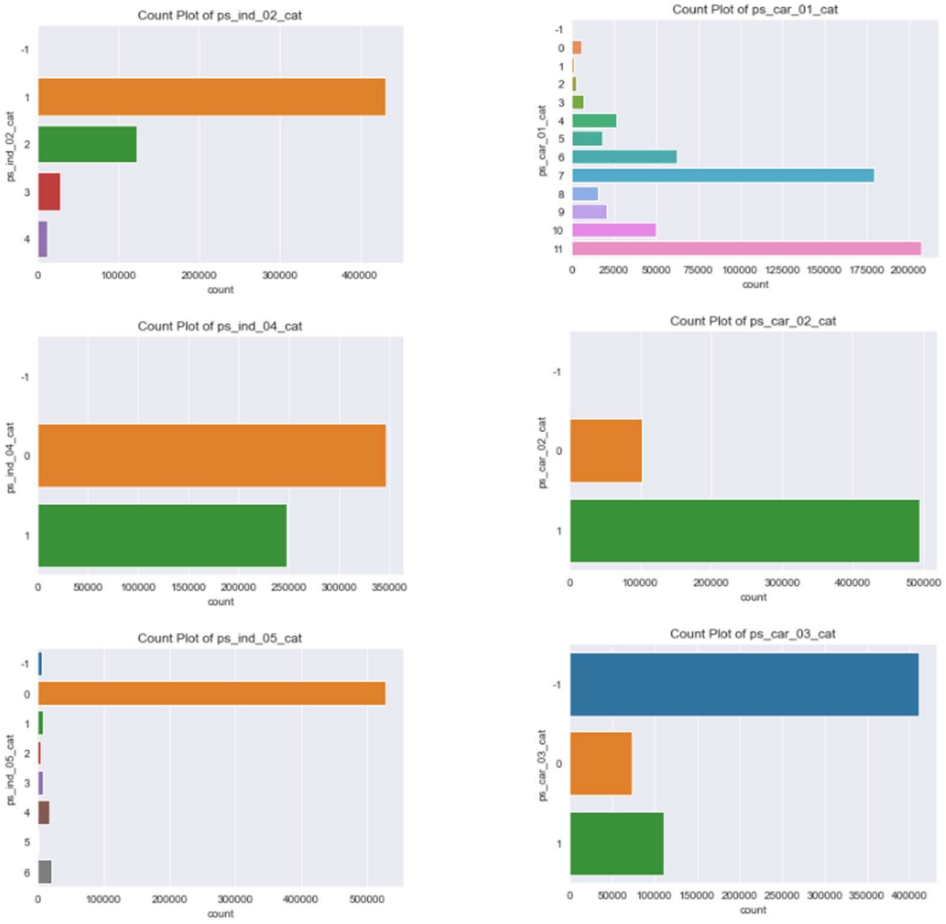
Figure: Histograms of numeric features

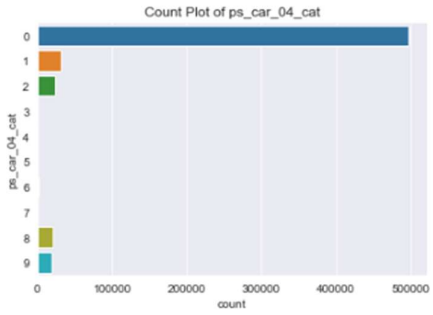
From the count plots of the categorical features (which are already coded and available as numeric values), we observe that:

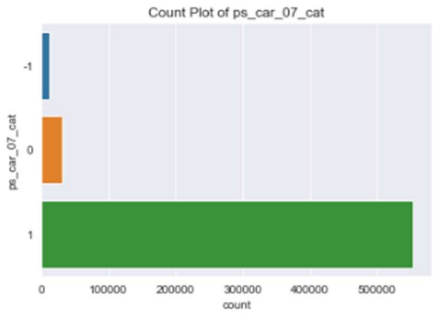
- The features – ps_ind_02_cat, ps_ind_05_cat, ps_car_01_cat, ps_car_04_cat, ps_car_06_cat, ps_car_07_cat, ps_car_10_cat, ps_car_10_cat have sparse classes.

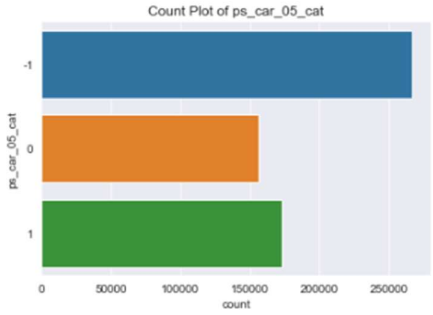
Also, as the feature names and values are already masked, so we cannot apply domain knowledge to combine a feature's two or more classes/categories into one class.

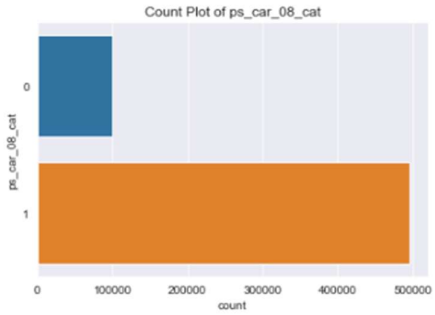
Figure: Scatter plots of categorical features

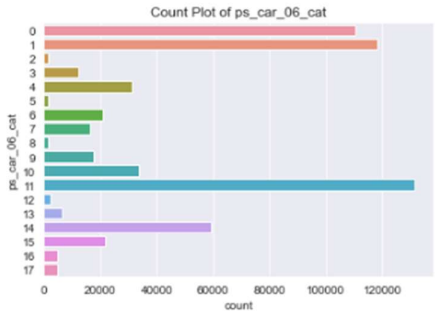


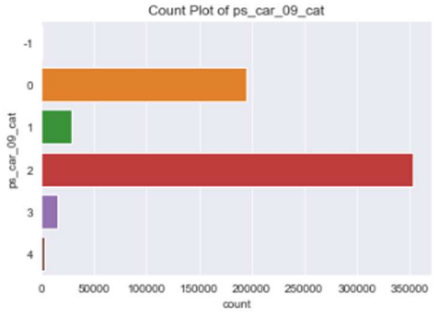


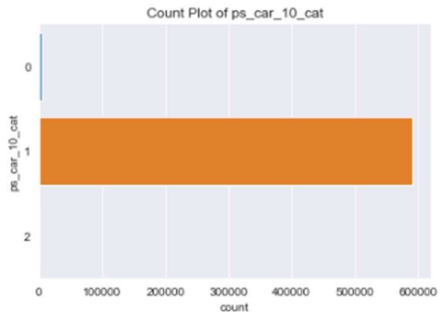


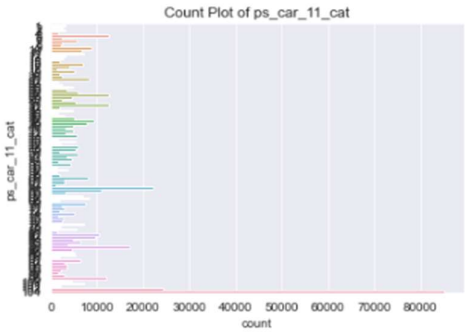












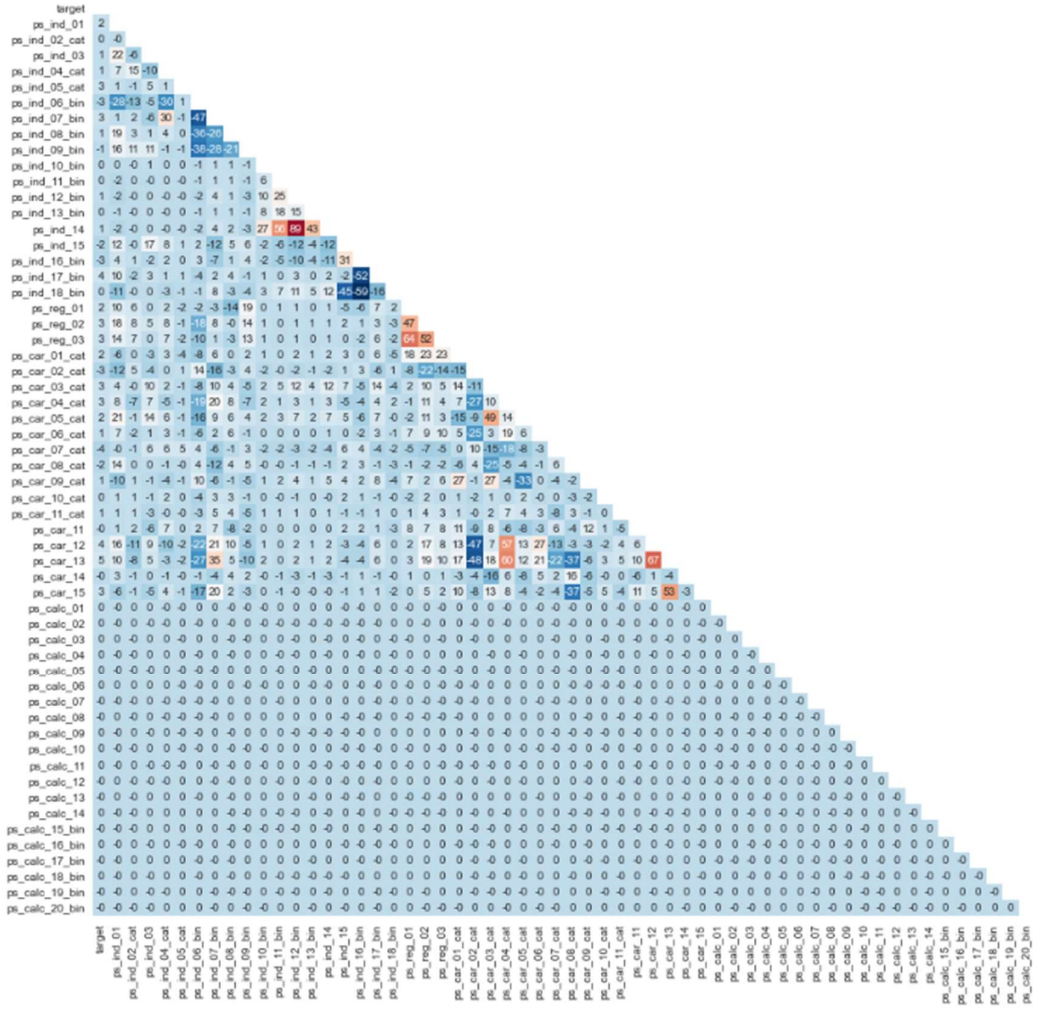
Correlations among various features

From the correlation grid of all features, we observe that:

- None of the features are strongly correlated with the 'target' variable
- Even, the other features do not have any strong correlation among each other, except two features (ps_ind_12_bin and ps_ind_14) which have correlation of 0.89 between them.

- All the calculated features, which have 'calc' in their names (ps_calc_01, ps_calc_02,...,ps_calc_20_bin) do not have ZERO correlation with all other features. Which may imply that they are pre-engineered features, in completely different dimensions, provided as part of the dataset.

Figure: Heatmap of correlation of all features



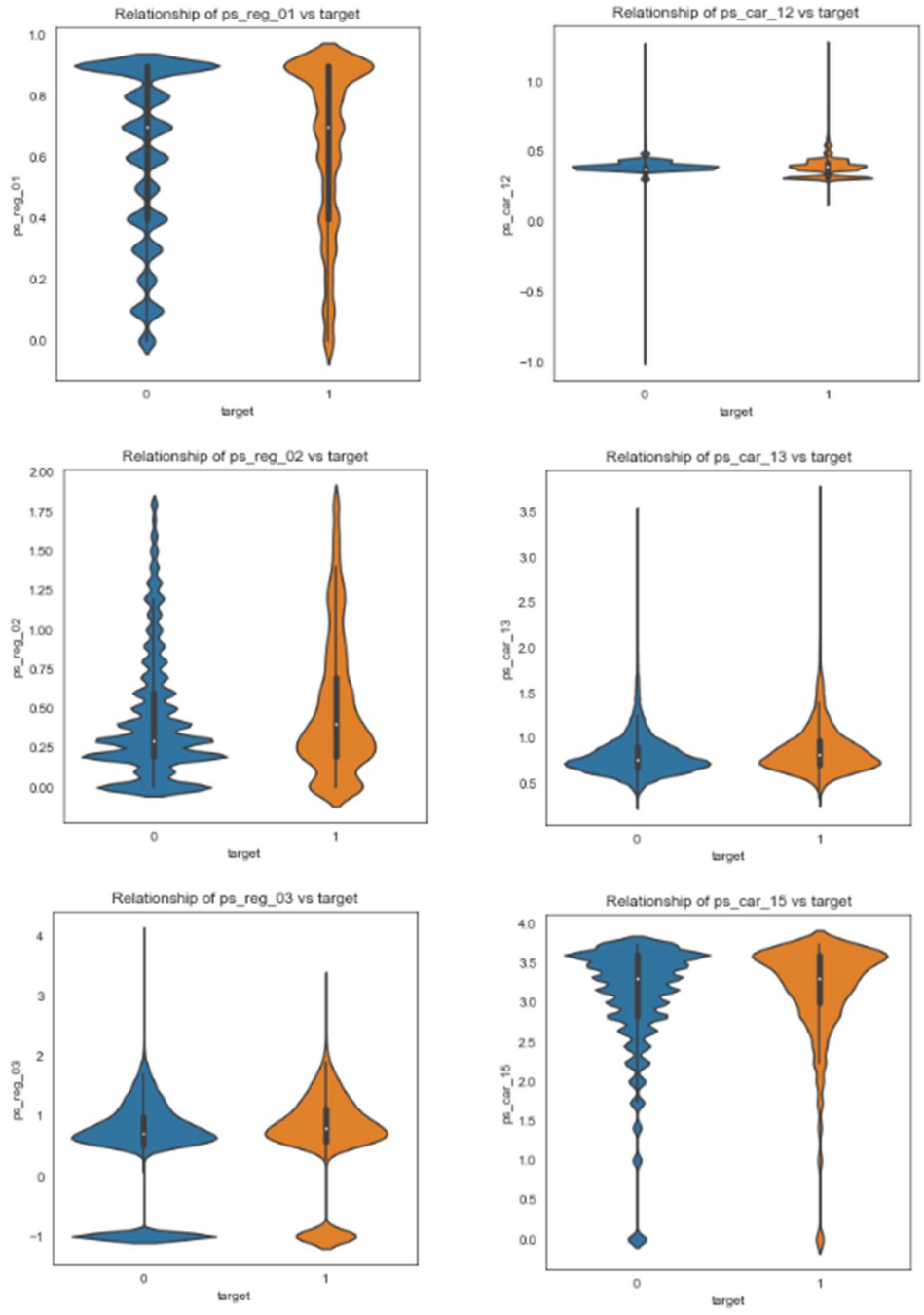
Segmentations of various features

Univariate segmentation

As part of univariate segmentation, we plotted violin plots of the the continuous numeric features (which have non-zero correlation with target variable) with the target variable.

From univariate segmentation, we observed similar results from the violin plot as in the correlation heatmap. We do not get any useful insight from the violin plots.

Figure: Violin plots of continuous numeric features vs target feature

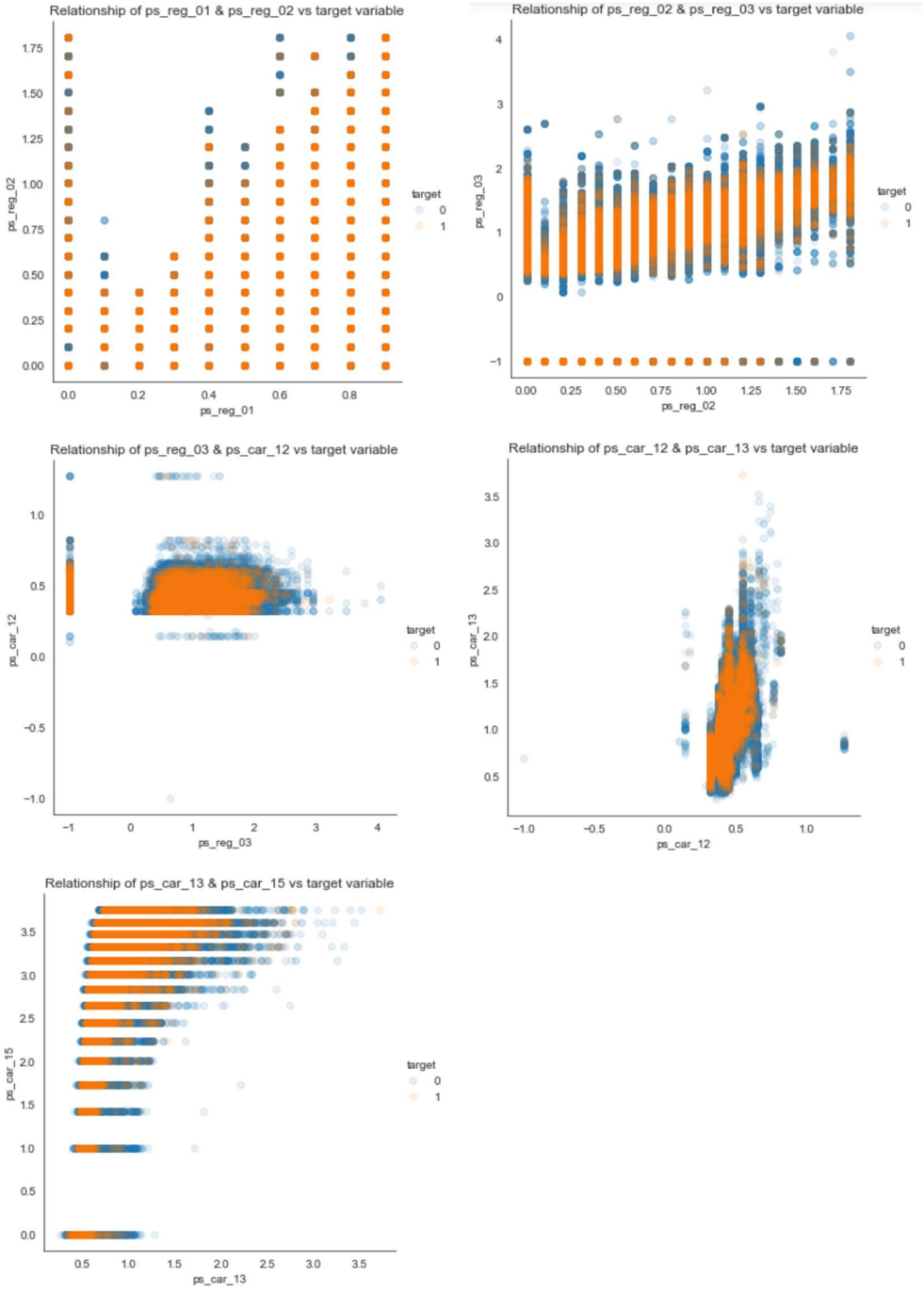


Bivariate Segmentation

As part of bivariate segmentation, we try to find insights, if any, from the combination of two continuous features with the 'target' variable, by plotting scatterplots of two continuous features segmented by 'target'.

However, we are unable to get any insight or clustering information about the relationship of the features with 'target'.

Figure: Scatterplots of Bivariate segmentation



Feature Selection

Our dataset has total of 58 features(including target variable). And, we saw that the features are not very correlated among themselves. Which brings us to the Feature Selection techniques to select and keep only important features in our dataset to avoid overfitting issues in our model building. We will use 2 features selection techniques:

1. Forward Feature Selection
2. Feature Selection using L1-regularization

The final list of selected features will be the combination of features selected from the above techniques.

Forward Feature Selection (FFS) using MLXtend library

Reference of FFS text: https://analyticsindiamag.com/a-complete-guide-to-sequential-feature-selection/

Forward Feature Selection is a Wrapper Method of feature selection, where a model is trained with a reduced number of features (i.e., a subset of features).

In Wrapper method of feature selction, the algorithm pushes a set of features iteratively in the model and in iteration the number of features gets reduced or increased. The stopping criteria of the iteration are to select the features which will give the best result in modelling. So at the end of the procedure of the wrapper method, the optimal set of features gets selected for the modelling.

In Forward Feature selection, features are sequentially added to an empty set of features until the addition of extra features till we reach the threshold number of features is reached.

We performed FFS on our dataset to select maximum 30 features.

The 30 features selected by FFS are: `'ps_ind_01', 'ps_ind_02_cat', 'ps_ind_03', 'ps_ind_04_cat', 'ps_ind_05_cat', 'ps_ind_06_bin', 'ps_ind_09_bin', 'ps_ind_15', 'ps_ind_16_bin', 'ps_ind_17_bin', 'ps_reg_01', 'ps_reg_02', 'ps_reg_03', 'ps_car_01_cat', 'ps_car_02_cat', 'ps_car_03_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_11', 'ps_car_12', 'ps_car_13', 'ps_car_14', 'ps_car_15', 'ps_calc_01', 'ps_calc_02', 'ps_calc_03'`

Feature Selection using L1-regularization using scikitlearn's SelectFromModel

Reference URL : *https://www.datatechnotes.com/2021/04/selectfrommodel-feature-selection.html*

In implementing Feature Selection using L1- regularization, ScikiltLearn's SelectFromModel class can be for extracting best features of given dataset according to the importance of weights. The SelectFromModel is a meta-estimator that determines the weight importance by comparing to the given threshold value.

SelectFromModel requires an estimator and we used L1-regularized LogisticRegression class for this purpose. An estimator model must have attributes to provide the indexes of selected data like 'get_support()' function. We defined the model and fit the model on X and y data. After the training, we get status of each feature data. To identify the selected features we use get_support() function and filter out them from the features list.

The 30 features selected by L1-regularization as: `'ps_ind_02_cat', 'ps_ind_04_cat', 'ps_ind_05_cat', '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_14', 'ps_ind_15', 'ps_ind_16_bin', 'ps_ind_17_bin', 'ps_ind_18_bin', 'ps_reg_01', 'ps_reg_02', 'ps_reg_03', 'ps_car_02_cat', 'ps_car_03_cat', 'ps_car_05_cat', 'ps_car_07_cat', 'ps_car_08_cat', 'ps_car_10_cat', 'ps_car_11', 'ps_car_12', 'ps_car_13', 'ps_car_15', 'ps_calc_01', 'ps_calc_03'`

Final List of Selected Features

Our final feature selection is the UNION of features selected by "Forward Selection" and "L1 regularization", which is 38 features.

So, we have final 38 input features selected for further stages of our machine learning workflow.

Model Building and Error Analysis

Model Building

Naïve Bayes Classifier (GaussianNB)

Gaussian Naive Bayes is a probabilistic classification algorithm based on applying Bayes' theorem with strong independence assumptions.

Bayes' theorem is also known as Bayes' Rule or Bayes' law, is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Evaluation of our Naïve Bayes model

1. We get cross-validated score of 0.946
2. TPR = 0.036, FPR = 0.02
3. AUROC = 0.613
4. Log Loss = 0.279
5. Normalized Gini Coefficient = 0.019

The Naïve Bayes model is our baseline model.

L1-regulaized Logistic Regression

Logistic regression is a statistical method that is used to model a binary response variable based on predictor variables. Logistic regression is very similar to linear regression as a concept and it can be thought of as a “maximum likelihood estimation” problem where we are trying to find statistical parameters that maximize the likelihood of the observed data being sampled from the statistical distribution of interest.

L1 regularized logistic regression an important tool in data science and is dedicated to solve sparse generalized linear problems. It is widely used for many classification problems, particularly ones with many features. This method avoids overfitting, by compressing the coefficients of the model by using the “absolute value of the coefficient function” as a penalty term. By adding L1 regularization to log-likelihood function of Logistic model, classification tasks based on the logistic regression model can be realized.

Evaluation of our L1-regularized Logistic Regression

1. We get cross-validated score of 0.964
2. True Positive rate = 0.586, False Positive Rate = 0.416

3. We get AUROC of 0.617 (Slight improvement as compared to our Naive Bayes model)
4. Log-Loss of 0.153 (Better than our Naive Bayes model)
5. Normalized Gini Coefficient = 0.172 (Much improved than Naive Bayes model)

L2-regularized Logistic Regression

L2-regularization adds the "squared magnitude" of the coefficient as the penalty term to the loss function.

In L2-regularization, the penalty term regularizes the coefficients of the model, and hence reduces the magnitudes of the coefficients that help to decrease the complexity of the model.

Evaluation of our L2-regularized Logistic Regression

1. We get cross-validated score of 0.964 (same as L1-Logistic Regression)
2. True Positive Rate = 0.586, False Positive Rate = 0.416 (same as L1-Logistic Regression)
3. We get AUROC of 0.617 (same as L1-Logistic Regression)
4. Log loss of 0.153 (same as L1-Logistic Regression)
5. Normalized Gini Coefficient = 0.172 (same as L1-Logistic Regression)

Elastic Net-regularized Logistic Regression

Elastic Net Logistic Regression is the variant of Logistic Regression where both L1 and L2 regularization is used to reduce the overfitting of the model. The ratio of L1 to L2 regularization can be controlled using the parameter 'l1_ratio'.

Evaluation of our Elasticnet-regularized Logistic Regression

1. We get cross-validated score of 0.964 (same as L2-Logistic Regression)
2. True Positive Rate = 0.586, False Positive Rate = 0.416 (same as L1-Logistic Regression and L2-Logistic Regression)
3. AUROC of 0.617 (same as L1 and L2-Logistic Regression)
4. Log Loss of 0.153 (same as L1 and L2-Logistic Regression)
5. Normalized Gini Coefficient = 0.172 (same as L1 and L2-Logistic Regression)

Decision Tree Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas

Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of the features of the given dataset.

Evaluation of our Decision Tree Classifier

1. We get cross-validated score of 0.964 (same as Logistic Regression)
2. True Positive Rate = 0.545, False Positive Rate = 0.411 (slightly less than the Logistic Regression models)
3. AUROC of 0.595 (less than the Logistic Regression models)
4. Log Loss of 0.155 (Logistic Regression models are slightly better than this)
5. Normalized Gini Coefficient = 0.136 (less than the Logistic Regression models)

Hence, we conclude that the Decision Tree Classifier's evaluation metrics are slightly poorer than the Logistic Regression models (L1, L2 and Elastic Net)

Summary

We found that all the Logistic Regression models (L1, L2 and Elastic-Net) performed the best and had the same metric values:

- AUROC of 0.617
- Log Loss of 0.153
- Normalized Gini Coefficient = 0.172

Advance Modelling and Feature Engineering

Model Building

Random Forests Classifier

A Random Forest is an ensemble of Decision Trees, generally trained via the bagging method, typically with `max_samples` set to the size of the training set.

With a few exceptions, a `RandomForestClassifier` has all the hyperparameters of a `DecisionTreeClassifier`, plus all the hyperparameters of a `BaggingClassifier` to control the ensemble itself.

The Random Forest algorithm introduces extra randomness when growing trees; instead of searching for the very best feature when splitting a node, it searches for the best feature among a random subset of features. The algorithm results in greater tree diversity, which trades a higher bias for a lower variance, yielding an overall better model.

Evaluation of our RandomForestsClassifer model

6. We get cross-validated score of 0.946
7. TPR = 0.577, FPR = 0.434
8. AUROC = 0.599
9. Log Loss = 0.162
10. Normalized Gini Coefficient = 0.145

We have Logistic Regression models performing better than our Random Forest model.

Gradient Boosting Decision Trees Classifier (GBDT)

GBDTs use boosting method to combine individual decision trees.

Boosting means combining a learning algorithm in series to achieve a strong learner from many sequentially connected weak learners. In case of gradient boosted decision trees algorithm, the weak learners are decision trees.

Each tree attempts to minimize the errors of previous tree. Trees in boosting are weak learners but adding many trees in series and each focusing on the errors from previous one make boosting a highly efficient and accurate model. Unlike bagging, boosting does not involve bootstrap sampling. Each time a new tree is added, it fits on a modified version of initial dataset.

Since trees are added sequentially, boosting algorithms learn slowly. In statistical learning, models that learn slowly perform better.

Evaluation of our GBDT Classifier

6. We get cross-validated score of 0.964
7. True Positive rate = 0.6, False Positive Rate = 0.428

8. We get AUROC of 0.626
9. Log-Loss of 0.153
10. Normalized Gini Coefficient = 0.173

Stacking Classifier

Stacking or Stacked Generalization is an ensemble machine learning algorithm.

It uses a meta-learning algorithm to learn how to best combine the predictions from two or more base machine learning algorithms.

The benefit of stacking is that it can harness the capabilities of a range of well-performing models on a classification or regression task and make predictions that have better performance than any single model in the ensemble.

We used Gaussian Naïve bayes, Decision Tree, Random Forest and GBDT as our base learners, and L1-regularized Logistic Regression as our meta learner in the Stacking Classifier.

Evaluation of our Stacking Classifier

6. True Positive Rate = 0, False Positive Rate = 0
7. We get AUROC of 0.508
8. Log loss of 0.156
9. Normalized Gini Coefficient = 0.004

Multi Layer Perceptron Classifier

The multilayer perceptron (MLP) is a feedforward artificial neural network model that maps input data sets to a set of appropriate outputs. An MLP consists of multiple layers and each layer is fully connected to the following one. The nodes of the layers are neurons with nonlinear activation functions, except for the nodes of the input layer. Between the input and the output layer there may be one or more nonlinear hidden layers.

We used Scikit-Learn library for training our MLP Classifier.

Our best MLP Classifier was found to be with the following hyperparameters:
`solver='sgd', max_iter=50, hidden_layer_sizes= (20, 10), alpha=0.05 and activation='tanh'`

Evaluation of our MLP Classifier

6. We get cross-validated score of 0.964
7. True Positive Rate = 0.617, False Positive Rate = 0.435
8. AUROC of 0.617
9. Log Loss of 0.153
10. Normalized Gini Coefficient = 0.173

Summary

As part of model building, we built the below models:

1. Naive Bayes Classifier (GaussianNB)
2. L1-regularized Logistic Regression
3. L2-regularized Logistic Regression
4. Elastic Net-regularized Logistic Regression
5. Decision Tree Classifier
6. Random Forests Classifier
7. Gradient Boosting Tress Classifier
8. Stacking Ensemble Classifier
9. Multi-Layer Perceptron Classifier

As part of model evaluation and error analysis, we evaluated/analysed their:

1. AUROC
2. Log Loss
3. Normalized Gini Coefficient

We found that all the **Logistic Regression models (L1, L2 and Elastic-Net) performed the best** and had the same metric values:

- AUROC of 0.617
- Log Loss of 0.153
- Normalized Gini Coefficient = 0.172

So, we finalize the **ElasticNet Logistic Regression** as our final model for this problem.

Deployment and Productionization

The final (winning) model – Elastic Net Logistic Regression

All the **Logistic Regression models (L1, L2 and Elastic-Net) performed the best** and had the same metric values:

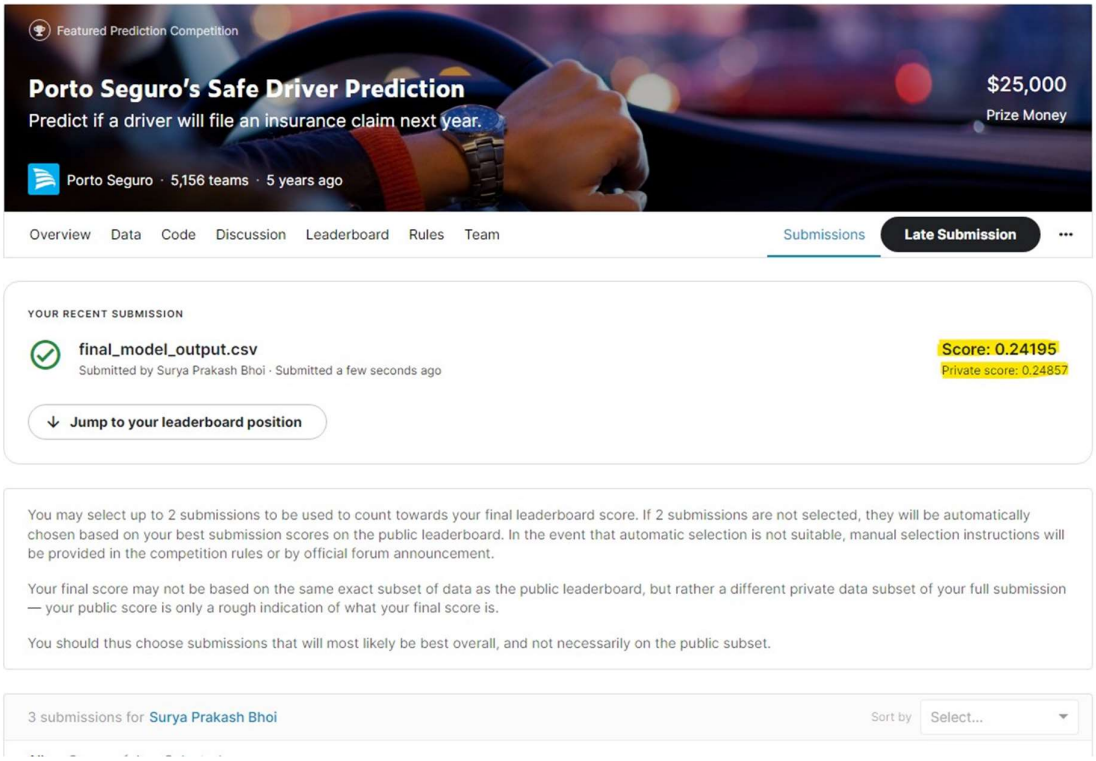
- AUROC of 0.617
- Log Loss of 0.153
- Normalized Gini Coefficient = 0.172

So, we finalize the **ElasticNet Logistic Regression** as our final model for this problem.

Evaluation using Kaggle's submission

Uploading the output file in kaggle submission gives a **private score = 0.24857**, while the **kaggle leader private score is 0.29698**. And **public score = 0.24195**, while the **kaggle leader public score is 0.29154**.

Rank for the private score of 0.24857 is **4227 out of 5158** submissions, which is in **81.9%**



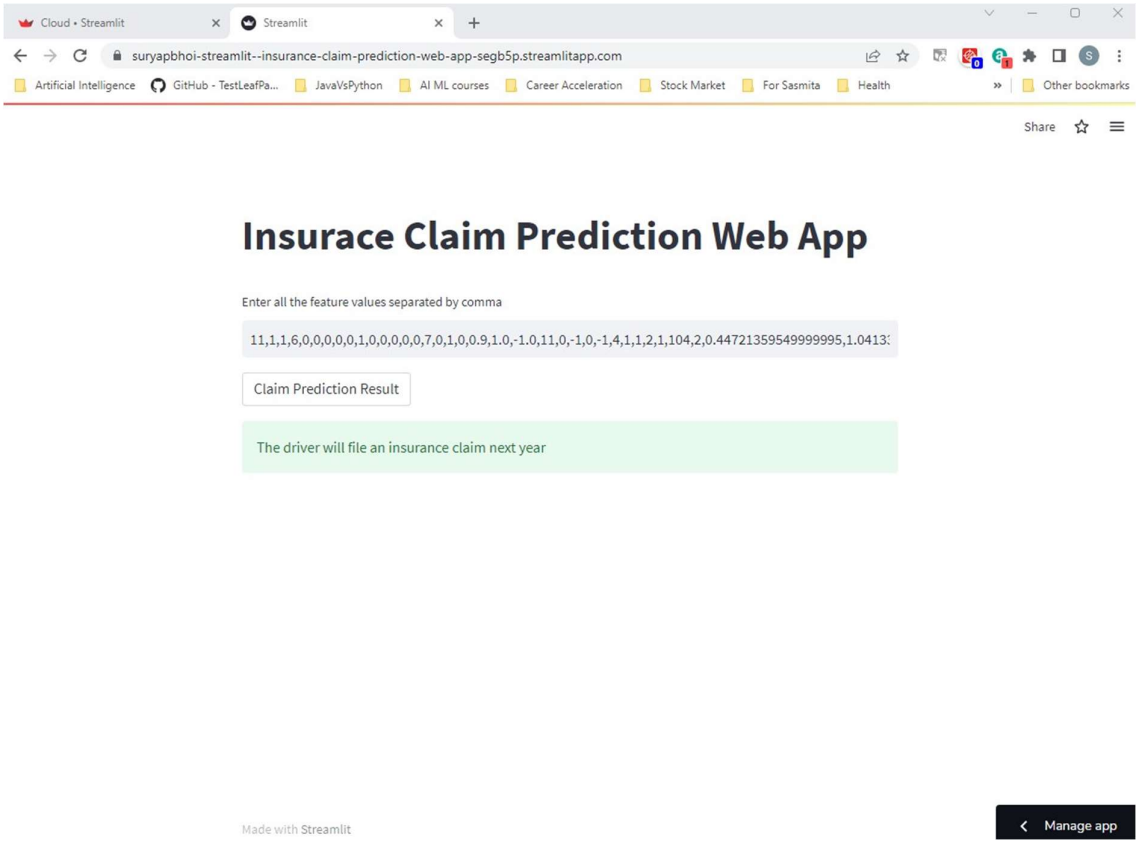
Kaggle score screenshot

Predictive system which predicts a single observation

This is a predictive system which performs the prediction for a single data point entered by the user directly into the web app, with field values separated by commas.
The web application was deployed using streamlit cloud platform.

Application URL: https://suryapbhoi-streamlit--insurance-claim-prediction-web-app-segb5p.streamlitapp.com/

Github URL: https://github.com/suryapbhoi/streamlit-insurance-claim-prediction-app



Screenshot of predictive system for a single observation

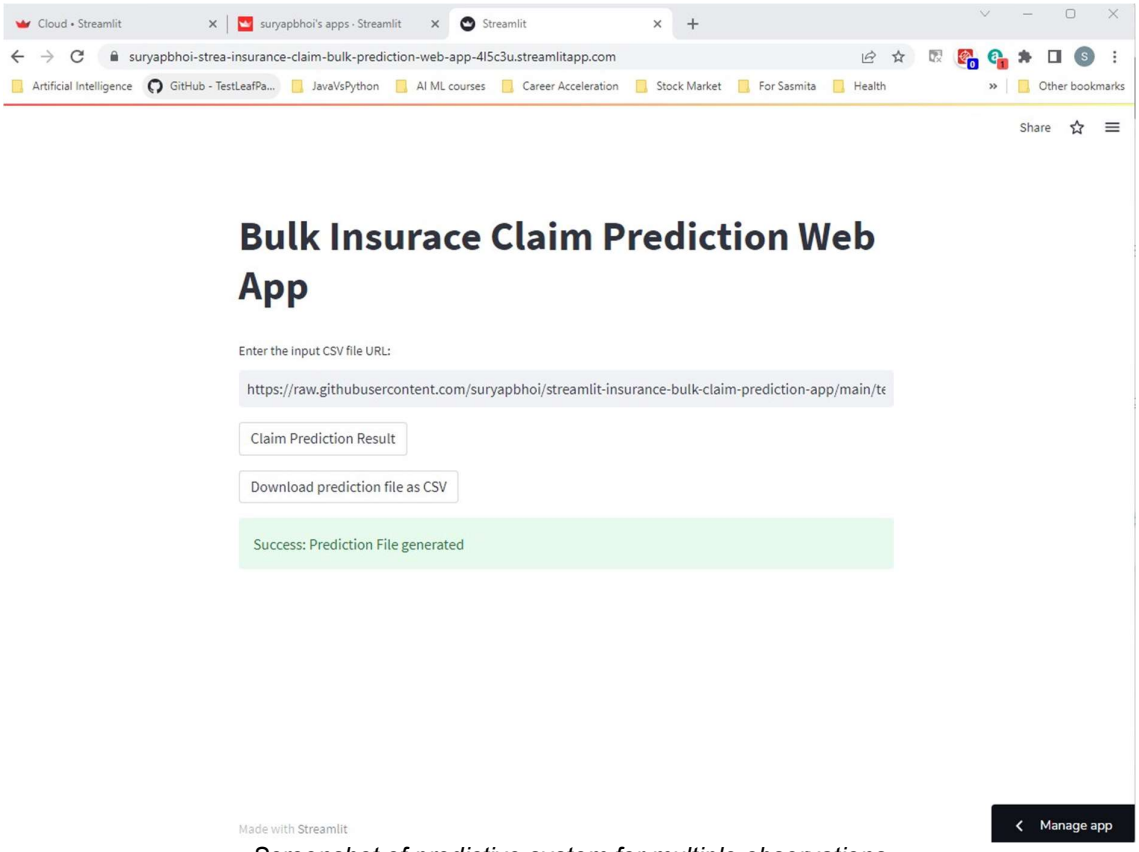
Predictive system which predicts more than one observation stored in csv file

This is a predictive system which performs the prediction in bulk for multiple data points stored in a csv file, which is passed into the web app.
The web application was deployed using streamlit cloud platform.

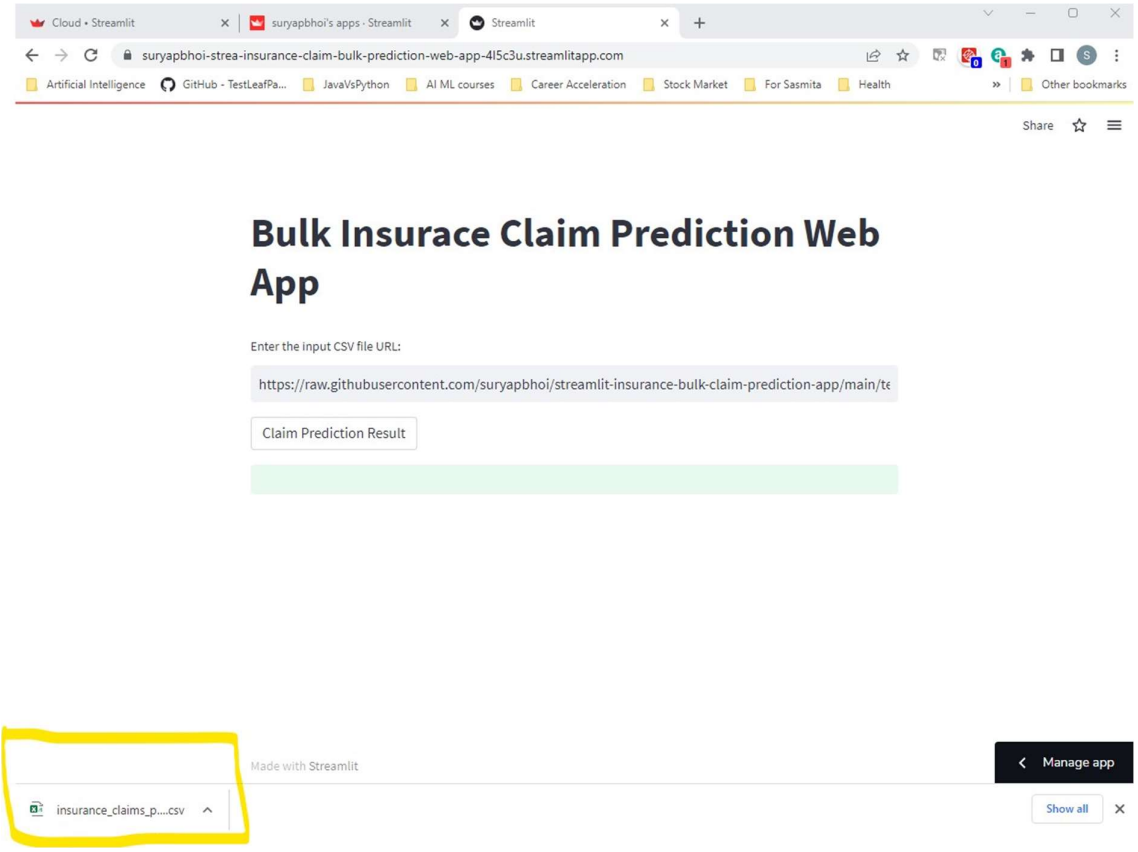
Application URL: https://suryapbhoi-strea-insurance-claim-bulk-prediction-web-app-4l5c3u.streamlitapp.com/

Github URL: https://github.com/suryapbhoi/streamlit-insurance-bulk-claim-prediction-app

Test file used: https://raw.githubusercontent.com/suryapbhoi/streamlit-insurance-bulk-claim-prediction-app/main/test_subset.csv



Screenshot of predictive system for multiple observations



Screenshot of predictive system downloadable output file

Github Repo of Project code

https://github.com/suryapbhoi/drivers-insurance-claim-prediction

Conclusion

We have successfully predicted whether “Insurance Claims” will be done by the insured next year. The Elastic Net Logistic Regression proved to be the best model with metric values:

- AUROC of 0.617
- Log Loss of 0.153
- Normalized Gini Coefficient = 0.172

References

[1] Xianzhi Liu and Qingquan, 2018. Song Safe Driver Prediction Base on Different Machine Learning Models.

[2] https://en.wikipedia.org/wiki/Logistic regression

[3] https://medium.com/analytics-vidhya/porto-seguros-safe-driver-prediction-a-machine-learning-case-study-1f2dd3f79d9e

[4] https://www.kaggle.com/code/durgancegaur/a-guide-to-any-classification-problem