

Porto Seguro's Safe Driver Prediction

Project Status Report

Hanlin He, Mingze Xu, Su Yang, Tao Wang

Department of Computer Science, The Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas

Email: {hxx160630, mxx160530, sxy161730, txw162630}@utdallas.edu

Abstract—This is the project status report of Porto Seguro's Safe Driver Prediction. Some of the content would be part of the final report.

problem, including *logistic regression*, *tensor-flow estimator (neural network)* and *random forest*.

I. INTRODUCTION

Machine learning is emerging in the insurance industry and is being applied across multiple areas including the interpretation of data, business operations and driver safety. One key application is claim prediction. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones. A more accurate prediction will allow insurers to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

In this report, we based on Kaggle's Featured Prediction Competition *Porto Seguro's Safe Driver Prediction* [1], conducted several experiments, compared different approaches' effectiveness to tackle the claim prediction

The organization of the following report is as follows. In section II, we will formally define the problem to solve and discuss the theoretical principle of the algorithm we used. Then we will analyze the data feature and our method of feature engineering in section III. After that, the experimental results are shown and analyzed in section IV. Finally, we will discuss related works and conclude the report.

II. PROBLEM DEFINITION AND ALGORITHM

A. Task Definition

A machine learning problem is defined as to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by

P , improves with experience E [8]. The claim prediction problem can be defined as follow:

E previous year's policy holders' information and whether or not a claim was filed for that policy holder.

T predicting the possibility that an auto insurance policy holder will file an insurance claim next year.

P the accuracies and Gini Coefficient was used to measure the effectiveness of models.

B. Algorithm Definition

We used logistic regression, neural network in our model building. In this section, we will discuss the theoretical foundation of these algorithms.

1) *Logistic Regression*: Logistic Regression is an approach to learning functions of the form $f : X \rightarrow Y$ [7] or in our case $P(Y|X)$ where Y is discrete-valued, and $X = \langle X_1 \dots X_n \rangle$ is any vector containing discrete and continuous variables. The parametric model assumed by Logistic Regression in the case where Y is boolean is:

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^n w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

One reasonable approach to training Logistic Regression is to choose parameter values that maximize the conditional data likelihood. We

also used regularization to reduce the overfitting problem. The penalized log likelihood function is as followed:

$$W \leftarrow \arg \max_W \sum_l \ln P(Y^l | X^l, W) - \frac{\lambda}{2} \|W\|^2$$

where the last term is a penalty proportional to the squared magnitude of W .

In general, the algorithm used gradient ascent to repeatedly update the weights in the direction of the gradient, on each iteration changing every weight w_i , beginning with initial weights of zero, according to:

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W)) - \eta \lambda w_i$$

where η is a small constant which determines the step size. The actual implementation of scikit learn library includes multiple solvers, such as Stochastic Average Gradient (SAG) descent, SAGA and Broyden-Fletcher-Goldfarb-Shanno (LBFGS).

2) *Neural Network*: Artificial neural networks (ANNs) [2], a form of connectionism, are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as “cat” or “no cat” and using the analytic results to identify cats

in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming.

III. FEATURE ANALYSIS AND ENGINEERING

The data comes in the traditional Kaggle form of one training and test file each: `train.csv` and `test.csv`. Each row corresponds to a specific policy holder and the columns describe their features. The target variable is named `target` here and it indicates whether this policy holder made an insurance claim in the past.

A. Data Overview

In total, there are 595212 training data instances and 892816 testing data instance. Since it's a competition, testing data had no `target` label. We cannot test our model accuracy based on testing data. In stead, we would use k-fold and cross validation on the training data to evaluate our model, which is discussed in more detail in section IV.

In the train and test data, features that belong to similar groupings are tagged as such in the feature names (e.g., `ind`, `reg`, `car`, `calc`). In addition, feature names include the postfix `bin` to indicate binary features and `cat` to indicate categorical features. Features without these designations are either continuous or ordinal. Feature count in each category and type are shown in table I.

TABLE I
FEATURE COUNTS IN EACH CATEGORY AND TYPE

| Type | Binary | Categorical | Numeric | Total |
|-------------------|--------|-------------|---------|-------|
| <code>ind</code> | 11 | 3 | 4 | 18 |
| <code>reg</code> | 0 | 0 | 3 | 3 |
| <code>car</code> | 0 | 11 | 5 | 16 |
| <code>calc</code> | 6 | 0 | 14 | 20 |

Although feature's categories are provided, the meaning of each feature remains unknown. Some participants have guessed the meaning of several features, for example the *binary* variables `ps_ind_06-10` are *one-hot encoded*, and `ps_car_13` might be car's mileage. Our experiment show that some of the assumptions are plausible, however, we did not rely on these information to build our model.

B. Distribution Analysis

Features' distribution for `ind`, `reg`, `car`, `calc` are shown as histograms in fig. 1, fig. 2, fig. 3 and fig. 4 respectfully.

From histograms, we may conclude the following:

- Some binary features are dominated by True (False), for example `ps_ind_06-10`.
- Some features' distribution are fairly even, for example `ps_calc_01`, `ps_calc_02` and `ps_calc_03`.

It might seem plausible to discard these features in training phase. But the deletion

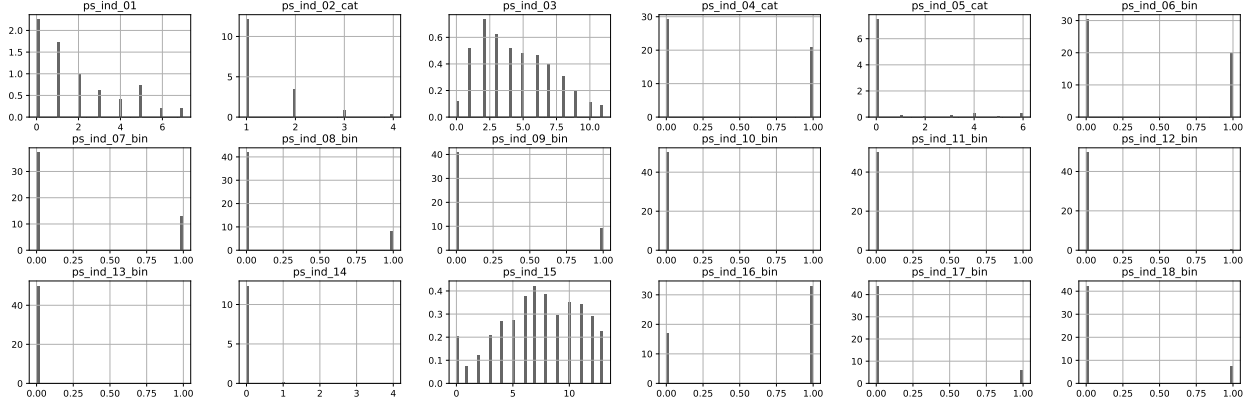


Fig. 1. Histogram for the `ind` Attributes.

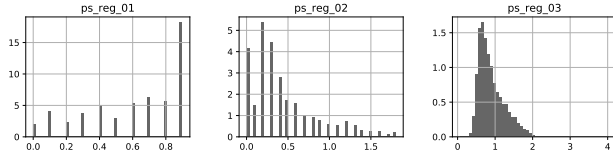


Fig. 2. Histogram for the `reg` Attributes.

requires more careful consideration. As mentioned above, `ps_ind_06-10` might be *one-hot encoded*, thus, all these four features would be naturally exclusive of each other.

C. Correlation Analysis

Features' Correlation within each category, i.e., `ind`, `reg`, `car`, `calc` are shown as histograms in fig. 5, fig. 6, fig. 7 and fig. 8 respectfully.

From correlation plot, we can conclude that strong correlation did not exist. It is not possible to conduct any dimension reduction based on correlations between features.

D. Missing Value Mechanism and Data Imputation

Values of -1 indicate that the feature was missing from the observation. Missing values count in each feature is shown in table II:

There are several approaches to handle missing values [9]:

- Deletion methods: cases with missing values are discarded, the analyses are restricted to cases that have complete data.
- Single imputation methods: imputes (i.e., *fills in*) the missing data with seemingly suitable replacement values. Mean, median and mode are the most popular averaging techniques to use [6].
- Multiple imputation methods: creates several copies of the data set, each containing different imputed values. This method has greater performance but is harder to implement.

From the table we can see that, most fea-

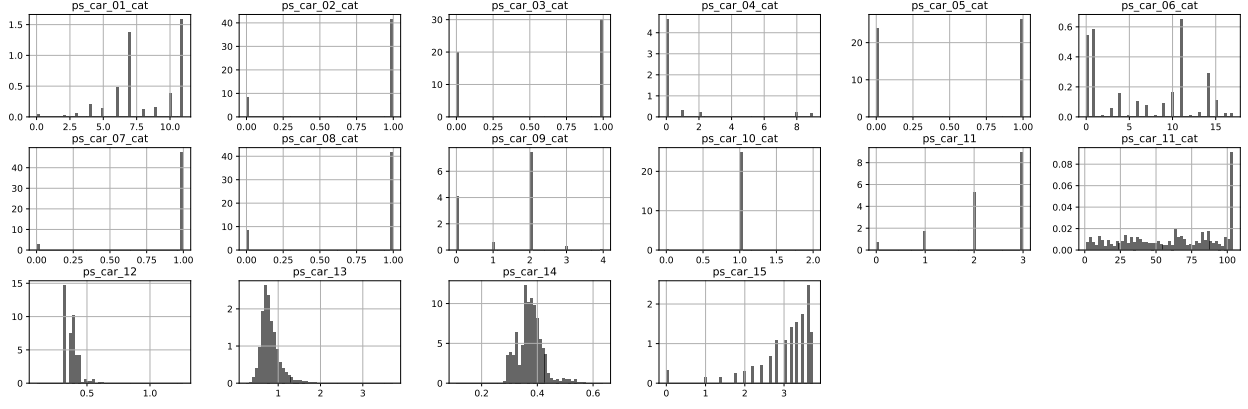


Fig. 3. Histogram for the `car` Attributes.

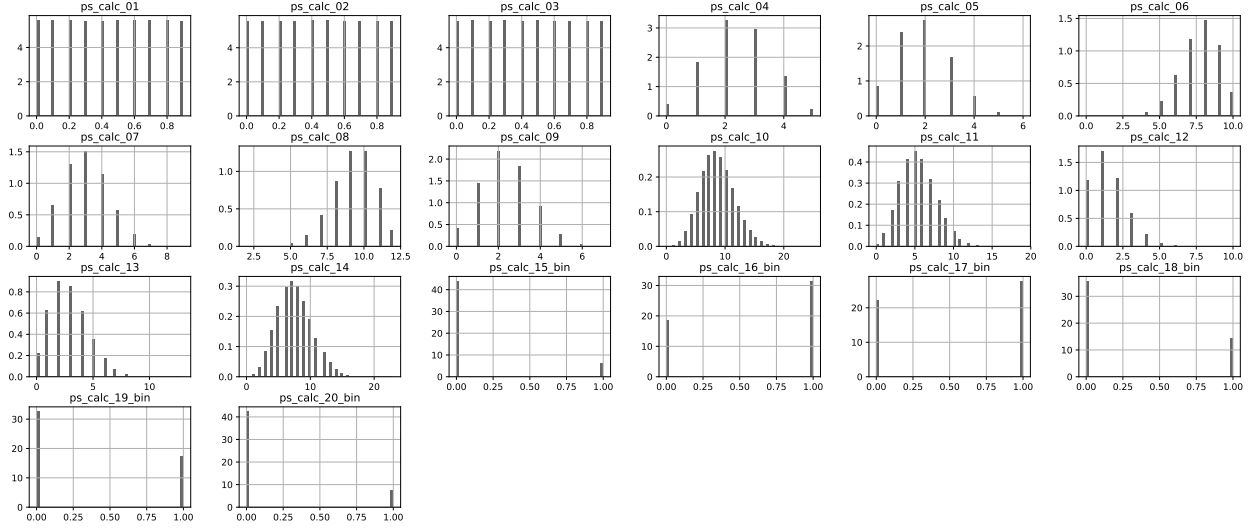


Fig. 4. Histogram for the `calc` Attributes.

tures have no missing value (only 13 features out of 51 have). Some features only have a small amount of instances with missing value (`ps_car_02_cat` and `ps_car_11` have 5 missing value and `ps_car_12` have only 1).

Based on these observation, we handled the missing values as follow:

- For features with a few missing values, delete those data instances.

- For categorical features, we treat the missing values a new category.
- For numeric features, we use the mean of the population as missing value.

IV. EXPERIMENTAL EVALUATION

A. Evaluation Metrics

We used per-class accuracy, AUC and Normalized Gini Coefficient to evaluate out model.

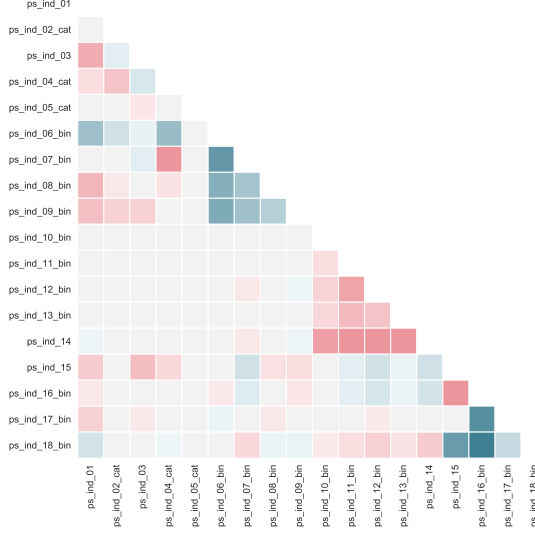


Fig. 5. Correlation for the `ind` Attributes.

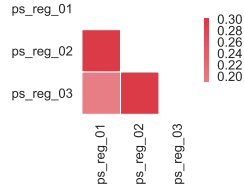


Fig. 6. Histogram for the `reg` Attributes.

1) *Per-Class Accuracy*: Accuracy simply measures how often the classifier makes the correct prediction. Its the ratio between the number of correct predictions and the total number of predictions (the number of data points in the test set) [10]:

$$accuracy = \frac{\# \text{ correct predictions}}{\# \text{ total data points}}$$

Per-class accuracy is the average of the accuracy for each class. By using per-class accuracy, we can have a better understanding of the model if the target class was dominated

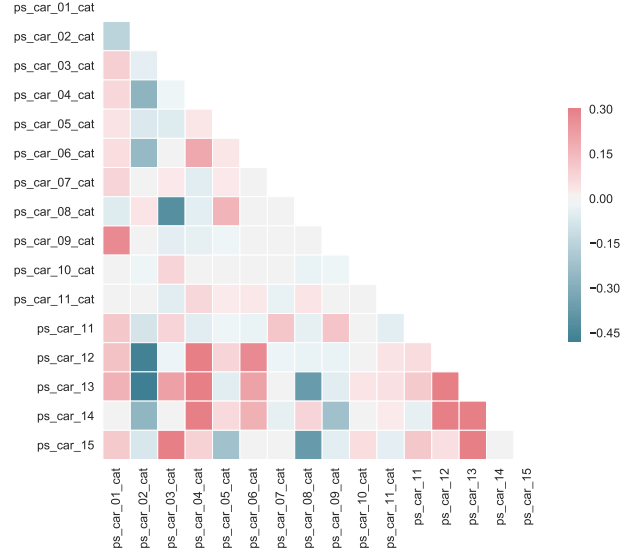


Fig. 7. Histogram for the `car` Attributes.

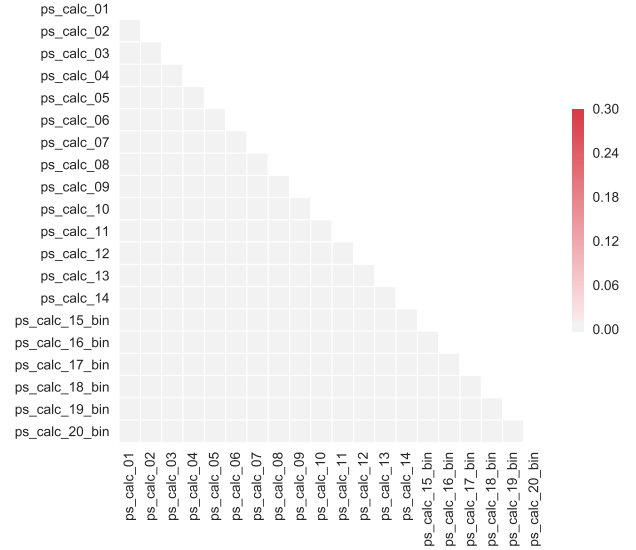


Fig. 8. Histogram for the `calc` Attributes.

by one label.

2) *Normalized Gini Coefficient and AUC*: The Normalized Gini coefficient, (named for the similar Gini coefficient/index used in Economics, which originally developed by Italian

TABLE II
MISSING VALUE COUNTS IN EACH FEATURE

| Feature Name | Count | Percentage |
|---------------|--------|------------|
| ps_ind_02_cat | 216 | 0.0363% |
| ps_ind_04_cat | 83 | 0.0139% |
| ps_ind_05_cat | 5809 | 0.9760% |
| ps_reg_03 | 107772 | 18.1065% |
| ps_car_01_cat | 107 | 0.0180% |
| ps_car_02_cat | 5 | 0.0008% |
| ps_car_03_cat | 411231 | 69.0898% |
| ps_car_05_cat | 266551 | 44.7825% |
| ps_car_07_cat | 11489 | 1.9302% |
| ps_car_09_cat | 569 | 0.0956% |
| ps_car_11 | 5 | 0.0008% |
| ps_car_12 | 1 | 0.0002% |
| ps_car_14 | 42620 | 7.1605% |

statistician and sociologist Corrado Gini [5]), measures the inequality among values of a frequency distribution (for example, levels of income) [3]. It is most commonly defined as twice the area between the ROC curve and the diagonal (with this area being taken as negative in the rare event that the curve lies below the diagonal).

AUC stands for area under the curve.

The normalized Gini coefficient and AUC are closely related. When using normalized units, the AUC is equal to the probability that a classifier will rank a randomly chosen positive

instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative') [4]. Elementary geometry shows that

$$Gini = 2 \times AUC - 1. \quad (1)$$

The competition used normalized Gini coefficient to measure participants' submission performance. In our experiment, we worked in terms of both Gini coefficient and AUC.

B. Evaluation Mechanisms

We use k-fold cross-validation as our validation technique.

V. CODING LANGUAGE

The main coding language is *Python 3.x* with following machine learning library:

- pandas.
- scikit-learn.
- tensorflow.

REFERENCES

- [1] Porto Seguros Safe Driver Prediction, 2017. [Online; accessed November 10, 2017].
- [2] Wikipedia: Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network, 2017. [Online; accessed November 10, 2017].
- [3] Wikipedia: Gini coefficient. https://en.wikipedia.org/wiki/Gini_coefficient, 2017. [Online; accessed November 10, 2017].
- [4] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.
- [5] Corrado Gini. *Variabilità e Mutuabilità*. 1912. Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche. C. Cuppini, Bologna.

- [6] Jacob Joseph. How to treat missing values in your data : Part I, 2016. [Online; accessed November 10, 2017].
- [7] Thomas M. Mitchell. *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression*. draft of February, 2016.
- [8] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.
- [9] Amanda N Baraldi and Craig K Enders. An introduction to modern missing data analyses. 48:5–37, 02 2010.
- [10] Alice Zheng. *Evaluating Machine Learning Models*. OReilly Media, Inc., 1 edition, 2015.