Capstone Project #1 Milestone Report

What is the problem?

Inaccuracies in car insurance company’s claim predictions can raise the cost of insurance for good drivers and reduce the price for bad ones. Nothing ruins the thrill of buying a brand-new car more quickly than seeing your new insurance bill. The sting’s even more painful when you know you’re a good driver. It doesn’t seem fair that you have to pay so much if you’ve been cautious on the road for years.

Who is my client?

My client is Porto Seguro, one of Brazil’s largest auto and homeowner insurance companies. While Porto Seguro has used machine learning for the past 20 years, they are looking to Kaggle’s machine learning community to explore new methods for further tailor their prices, and to hopefully make auto insurance coverage more accessible to more drivers.

Dataset description

Porto Seguro has provided two large dataset, one of training data and one of test data. Both dataset shares 57 features columns, an ID column, and training data has an extra target column that will either be 0 or 1, where 1 means driver did file for insurance claim the next year, and 0 means driver did not file for insurance claim. Training data contains 595212 rows and test data contains 892816 rows. Both train and test dataset are well formed, there are no Null values, however there are missing values, which are nicely marked by -1s. There are couple types of features:

Binary: type int64, has a suffix of \_bin, 1 or 0 to denote yes or no

Category: type int64 and has suffix of \_cat, integer value to denote which category

Other: type int64 or float64 and doesn’t have any suffix, continuous values of the specific feature

Other potential datasets

It is rather difficult to incorporate any other dataset, the reason is the dataset provided by Porto Seguro chose to give only code names for the features, thus, even if I was to incorporate other insurance claim related datasets, it would be hard to cross join the datasets together.

Initial findings

For data inference, my goal is to find any correlation between features and target. To do this, I first separated out the training dataset into one set of data where the target is 1 (true set) and another set where target is 0 (false set). Then I setup functions that would compute and plot the ECDF on both the true set and false set of a single feature. Then using the functions setup, I plotted the ECDF of each feature.

|  |  |
| --- | --- |
|  |  |

Next, I went through each individual carefully, and observed that certain features have different ECDF plots between true set and false set.

['ps\_ind\_01','ps\_ind\_03','ps\_ind\_05\_cat','ps\_ind\_06\_bin','ps\_ind\_07\_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\_cat','ps\_car\_12','ps\_car\_13','ps\_car\_15']

Here is the set of features that have different ECDFs. This helps reduces the features which I need to worry about down to 24.

Next, I ran Pearson correlation coefficient between quantitative features and the target, but the resulting Pearson r values are all relatively small (largest magnitude is 0.053). Then I ran chi-square test on nominal features (binary and categorical). To conduct the test, I first tallied up the number of occurrences of each of the unique combination between the feature and target. Then I used the set where target is 0 as the expected value and set where target is 1 as the observed value to compute chi square. There is a noticeable difference between features that had similar ECDF to those that didn’t. For example, ps\_ind\_11\_bin has similar ECDF plot:

chi2: 2.44839164851

p: 0.117645355928

dof: 1

expected: [[ 2.16572973e+04 3.67026505e+01]

[ 5.72547703e+05 9.70297350e+02]]

whereas ps\_ind\_06\_bin which has a different ECDF plot has much higher chi square and lower p value.

chi2: 688.748032799

p: 8.36655285187e-152

dof: 1

expected: [[ 13152.15971452 8541.84028548]

[ 347699.84028548 225818.15971452]]