DS-GA 1001 Project Final Report

Team members: Qintai Liu (ql819), Yicheng Pu (yp653), Yi Zhou (yz4525)

Team name: Whatever

Business Understanding

With the development of the transportation, more and more incidents and injuries took place on demand-response vehicles. To mitigate the loss from incidents, the industry of insurance emerges and gradually welcomed by every family with automobile. However, the insurance industry market is very competitive. Almost every insurance company could supply same quality services, so price becomes one of the most important incentives for customers to decide which company they want to purchase. Safe drivers are unwilling to pay much for their cautious driving history. Insufficient charging of bad drivers would bring loss to the car insurance company. Therefore, accuracies in car insurance company's claim predictions is very necessary for customers and should reduce the price for good drivers and raise the cost of unsafe ones. Moreover, in the perspective of the car insurance company, a reasonable pricing depending on the prediction is also irreplaceable. It does not only embody the concern of the company to their customers, but also is a brand promotion of their professional skills, which helps them stand out in the market.

The prediction model we want to form requires the driving experience in the past year and would return a probability of the driver being in an incident in the next year. According to the result of the model, the car insurance company could decide if it is necessary to raise of reduce the price of car insurance of the driver in the next year. Driver with high score would has

a higher chance of being charged more in the next year, while driver with low score are possible to have lower insurance fee.

Data Understanding and Preparation

We obtained data from kaggle[1].

There are total two data sets, which are train.csv and test.csv. The train data set has 59 columns, which is composed of 57 features and id and target variable, and 43388 rows, each of which represents a customer. The target variable is a binary data type. It's equal to one if the driver initiated an auto insurance claim and otherwise it's equal to zero.

The test data has the same amount of features as the train data, but doesn't contain target variable, which the project's goal to predict. The project tries to predict the probability that a driver will initiate an auto insurance claim in the next year based on the data provided in the test data.

After the prediction for the target variable is maken, the prediction will be written in submission file and kaggle will evaluate it.

The characteristics of features:

- 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.

[1] https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data

- Features without these designations are either continuous or ordinal.
- Values of -1 indicate that the feature was missing from the observation.[2]

Among of 57 features, there are

Feature type	Ordinal(int)	binary	float	categorical
count	16	17	10	14

Find the features with missing value

```
Variable ps_ind_02_cat has 216 records (0.04%) with missing values
Variable ps_ind_04_cat has 83 records (0.01%) with missing values
Variable ps_ind_05_cat has 5809 records (0.98%) with missing values
Variable ps_reg_03 has 107772 records (18.11%) with missing values
Variable ps_car_01_cat has 107 records (0.02%) with missing values
Variable ps_car_02_cat has 5 records (0.00%) with missing values
Variable ps_car_03_cat has 411231 records (69.09%) with missing values
Variable ps_car_05_cat has 266551 records (44.78%) with missing values
Variable ps_car_07_cat has 11489 records (1.93%) with missing values
Variable ps_car_09_cat has 569 records (0.10%) with missing values
Variable ps_car_11 has 5 records (0.00%) with missing values
Variable ps_car_12 has 1 records (0.00%) with missing values
Variable ps_car_14 has 42620 records (7.16%) with missing values
In total, there are 13 variables with missing values
```

From the above result, we can see ps_car_03_cat and ps_car_05_cat have a large proportion of records with missing values.

Descriptive statistic for float variables

[2] https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data

	ps_reg_01	ps_reg_02	ps_reg_03	ps_car_12	ps_car_13	ps_car_14	ps_car_15	ps_calc_01	ps_calc_02	ps_calc_03
count	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212
mean	0.61099138	0.43918436	0.55110184	0.37994482	0.81326468	0.27625627	3.06589944	0.44975639	0.44958922	0.44984879
std	0.28764262	0.40426429	0.79350577	0.05832696	0.2245881	0.35715403	0.73136623	0.28719809	0.28689348	0.28715299
min	0	0	-1	-1	0.25061907	-1	0	0	0	0
25%	0.4	0.2	0.525	0.31622777	0.67086659	0.33316663	2.82842712	0.2	0.2	0.2
50%	0.7	0.3	0.72067677	0.37416574	0.7658113	0.36878178	3.31662479	0.5	0.4	0.5
75%	0.9	0.6	1	0.4	0.9061904	0.39648455	3.60555128	0.7	0.7	0.7
max	0.9	1.8	4.03794502	1.26491106	3.720626	0.6363961	3.74165739	0.9	0.9	0.9

Insight gained from the statistic:

The calc variables seem to have the maximum is 0.9 and all of them have very similar distributions

Descriptive statistic for Ordinal variables

	ps_ind_01	ps_ind_03	ps_ind_14	ps_ind_15	ps_car_11	ps_calc_04	ps_calc_05	ps_calc_06	ps_calc_07	ps_calc_08	ps_calc_09	ps_calc_10	ps_calc_11	ps_calc_12	ps_calc_13	ps_calc_14
count	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212
mean	1.90037835	4.42331808	0.01245103	7.29992171	2.34607165	2.37208087	1.88588604	7.68944511	3.00582314	9.22590438	2.33903382	8.43359005	5.44138223	1.44191817	2.87228752	7.53902643
std	1.98378912	2.69990196	0.12754497	3.5460421	0.83254781	1.11721894	1.13492707	1.33431222	1.41456387	1.45967195	1.24694916	2.90459729	2.33287124	1.20296253	1.69488686	2.74665164
min	0	0	0	0	-1	0	0	0	0	2	0	0	0	0	0	0
25%	0	2	0	5	2	2	1	7	2	8	1	6	4	1	2	6
50%	1	4	0	7	3	2	2	8	3	9	2	8	5	1	3	7
75%	3	6	0	10	3	3	3	9	4	10	3	10	7	2	4	9
max	7	11	4	13	3	5	6	10	9	12	7	25	19	10	13	23

Descriptive statistic for binary variables

	target	ps_ind_06_b	ps_ind_07_b	ps_ind_08_b	ps_ind_09_b	ps_ind_10_b	ps_ind_11_b	ps_ind_12_b	ps_ind_13_b	ps_ind_16_b	ps_ind_17_b	ps_ind_18_b	ps_calc_15_t	ps_calc_16_t	ps_calc_17_t	ps_calc_18_t	ps_calc_19_t	ps_calc_20_bi
count	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212	595212
mean	0.03644752	0.39374206	0.25703279	0.16392143	0.18530372	0.00037298	0.00169183	0.00943865	0.00094756	0.66082337	0.12108123	0.15344617	0.12242697	0.62784016	0.55418238	0.28718171	0.34902354	0.15331848
std	0.18740105	0.48857922	0.436998	0.37020457	0.38854409	0.01930901	0.04109714	0.09669323	0.03076793	0.47343027	0.32622192	0.36041734	0.32777856	0.4833811	0.49705602	0.45244748	0.47666182	0.36029452
min	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25%	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50%	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0
75%	0	1	1	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0
max	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Insight gained from the statistic:

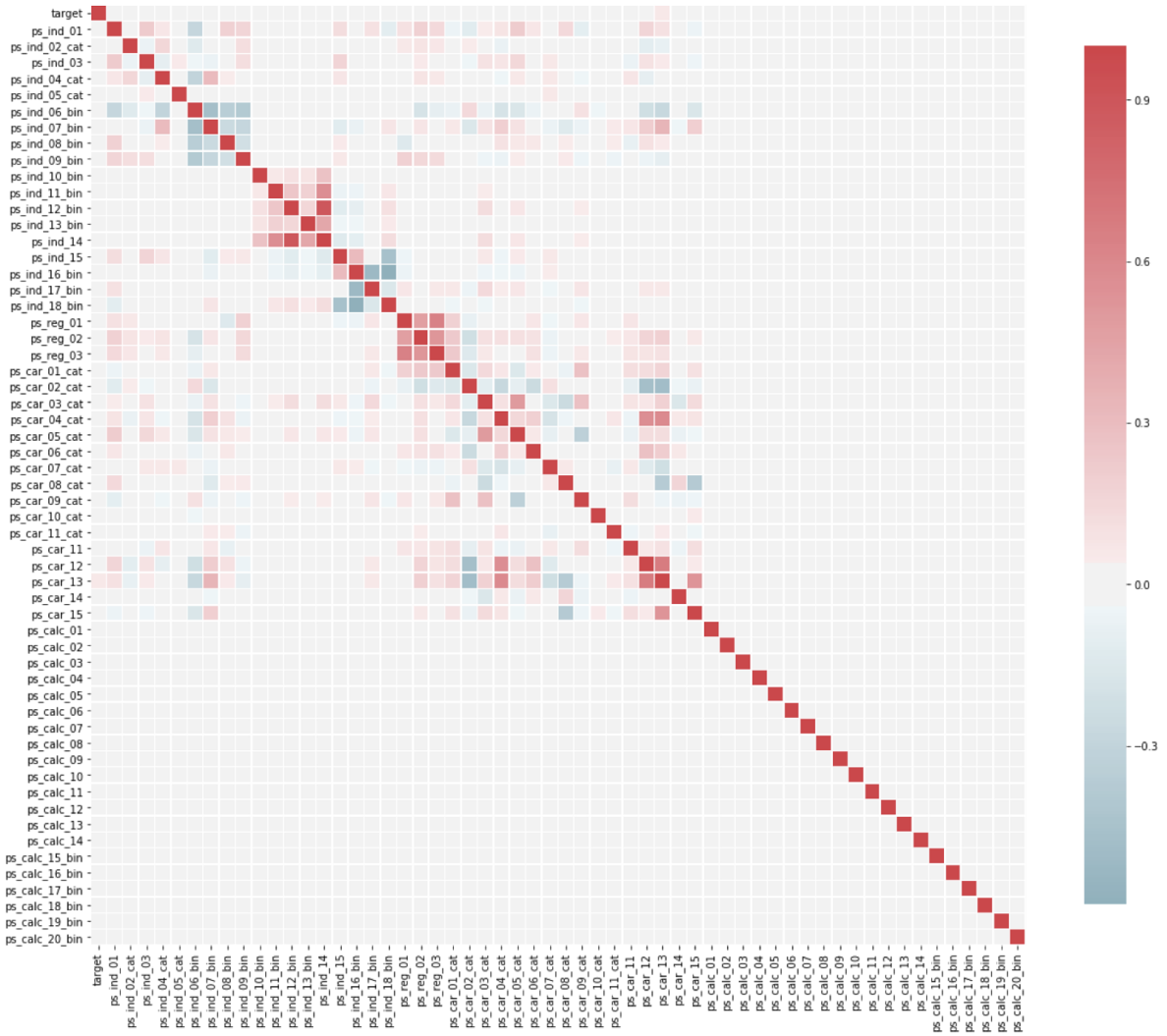
The target value equal to one only account for 0.003645, which is strongly imbalanced.

Checking the cardinality of the categorical variables

```
Variable ps_ind_02_cat has 5 distinct values
Variable ps_ind_04_cat has 3 distinct values
Variable ps_ind_05_cat has 8 distinct values
Variable ps_car_01_cat has 13 distinct values
Variable ps_car_02_cat has 3 distinct values
Variable ps_car_03_cat has 3 distinct values
Variable ps_car_04_cat has 10 distinct values
Variable ps_car_05_cat has 3 distinct values
Variable ps_car_06_cat has 18 distinct values
Variable ps_car_07_cat has 3 distinct values
Variable ps_car_08_cat has 2 distinct values
Variable ps_car_09_cat has 6 distinct values
Variable ps_car_10_cat has 3 distinct values
Variable ps_car_11_cat has 104 distinct values
```

From the result, we can see ps_car_11_cat has many distinct values.

Correlation coefficient matrix



From the correlation coefficient matrix, we find all of the calc variables has 0 correlation coefficient, which means these variables may are not related to other variables. So when building the model, we could consider whether to drop these variables or not.

Modeling

Evaluation Metric:

The evaluation metric used by the competition host is Gini coefficient. The Gini coefficient is calculated by firstly sorting the prediction scores, then compare the corresponding real values of sorted prediction with perfect prediction. The advantage of using Gini coefficient is that this metric totally depends on the ranking of prediction, but not specific values. For example, if the real value is [1,0,1,0,1], then prediction [0.1,0.2,0.3,0.4,0.5] has the same Gini coefficient with prediction [1,2,3,4,5] . Practically, we would also normalize the Gini coefficient with the max Gini coefficient, so in the last we would use normalized Gini coefficient as evaluation metric.

Baseline model

Intuitively, to solve a problem with binary target value, logistic regression would be our first choice for a baseline model. Even though the dataset has pretty high dimension and very unbalanced target distribution, which means that logistic regression, as a statistical model, would not perform well, we would still like to use it as a baseline model.

We dropped 2 columns with pretty high volume of missing values, since we believe that the missing part is too noisy for our model. For the rest missing values, we replaced them with different metrics: if the variable is ordinal, we would replace them with their modes; otherwise, we would replace them with their mean values. We then converted all categorical variables to dummy variables, and consider the missing categorical value as another category. We also drop all columns with variance lower than 0.01, since that means those predictors do not contain much information.

Then we performed 10-fold cross validation on weighted logistic regression different regularization values (C value) to pick the best C. The best performance is 0.185, with C=, the score is not good enough compare with other models on Kaggle, however, this is only a simple prototype and we could use it as a baseline model.

Analysis of possible algorithms

After doing logistic regression, we started to look for more complex and powerful models to solve this problem. Considering that this is a high-dimensional dataset, which contains more than 100 predictors after our feature engineering, we felt that KNN and SVM might be good candidates. This time we did even more feature engineering:

1. Down sampling the data
2. Dropping two variables with missing values
3. Replacing other missing values with mode if the missing fraction is low.
4. Replacing other missing values with mean if the missing fraction is high.

5. Adding dummy variables to all categorical data except ‘ps_car_11_cat’ since it has more than 100 categories.

6. Adding 2-degree polynomial features for all continuous features.

7. Using a Random forest to fit the model, then use its importance to drop half of the features.

After we finished the feature engineering, we firstly tried with SVM, unfortunately, the SVM costs too much computation power and running is so long that we need to switch to another approach.

For KNN, we did 5-fold cross validation with different K values. The Best performance is 0.194 with k=700. The Gini coefficient was significantly improved, but it is still not enough. We started to consider that whether dropping half of the features would really help and if adding ‘ps_car_11_cat’ back could further improve the model. Also when we looked at the correlation coefficient heat-map, we found that ‘calc’ variables are uncorrelated with target and other variables, so we decided to drop all ‘calc’ variables instead of dropping half of variables selected by random forest. We also add dummy variables for ‘ps_car_11_cat’. This time we got a best performance of 0.20549 with k=700.

Is 0.20549 the limit? We have seen a lot of higher scores on Kaggle so we are sure that there must be some room for improvement.

Final model

After doing some research on Kaggle, we found that gradient tree boosting (GTB) are pretty popular[3], so we decided to have a try on that. At the very beginning, we did not do any feature engineering except downsampling the data and dropping 2 predictors with high volume of missing values. Surprisingly, such an out-of-box model returned a Gini coefficient of 0.192, which implies that GTB could be a powerful model. Then we started doing feature engineering:

1. Dropping 4 predictors with high volume of missing values.
2. Adding dummy variables to all categorical data.
3. Drop all 'calc' variables.
4. Replacing other missing values with mode.

For down sampling, since GTB is a tree-based model, we felt it might be more robust than other models to deal with unbalanced data, also because the testing data is also unbalanced, we decided to not to do down sampling this time. Since GTB has too many parameters, we did not do grid search, but randomly test different combinations of parameters. In the end, using 5-fold cross validation, we got a Gini coefficient of 0.282, a huge leap!

Then we started to look at missing values. We did not thoroughly test those variables with high volume of missing values before we dropped them out, so we added them back and add another dummy variable to indicate whether the value is missing. This time the Gini coefficient is 0.290, another step forward!

[3] http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Deployment

Our model is to predict if the driver would be a safe driver in the next year. The result of prediction recommends the car insurance company to adjust the driver's insurance fee. If the result is closer to 1 than 0, then the company should raise some amount of insurance fee to avoid loss. If the result is closer to, which means that the driver is more likely to be a safe driver in the next year, then the insurance company should consider whether they could reduce the fee or keeping the current price. However, the precise amount of money the company would like to adjust could not be solved in this problem. Therefore, to better improve the pricing standards, the next objective of the car insurance company should be building another model to calculate the exact number they want to change with the input of prediction from our model.

The company should keep track of the predicted rate that a customer will initiate an auto insurance claim and the actual result about whether this customer initiate claim or not. The company should compare whether a customer who is predicted to be safe driver with those who is predicted to be risky driver to see whether the prediction matches the factual result.

The car insurance company has the obligation to protect customers' privacy, so they need to encode feature names of customer's information. Besides that, if the company has the need to public some statistics, they must add some noise on the original data to avoid information disclosure.

One problem the company might have to deal with is the equilibrium over races and genders, if those data is included. Even if the model generates high prediction score for specific

gender or races, it is not acceptable to use the score in practice. The company should avoid such kind of issues and exclude those data during training.

Since the our model is built on the current data, but the world is always changing, our model is not that “solid”. For example, if the automobile market promotes a new car model, and the dataset does not contain user information about the new car model, then the prediction made by our model might deflect from the real data. To mitigate similar problems, the company should follow up car news and investigate local transportation information on time, and update their model once they get news.

Appendix

Contributions

Our team worked together for most of the time, including model building and testing, but team members have specific contribution on suggesting different models and methods.

Qintai Liu: Suggested using KNN and SVM model, Created multiple feature engineering methods (Random Forest selection, dummy variable, etc.).

Yicheng Pu: Suggested using Gradient Boost Tree, adding missing value back to the model and dropping 'calc' variables.

Yi Zhou: Suggested using Logistic Regression, did initial data analysis and preprocessing (Dropping variables, low variance selection, etc.).

Our work has been uploaded to github: https://github.com/YichengPu/ps_insurance