Machine Learning Engineer Capstone Project: Porto Seguro's Safe Driver Prediction

Miao Tian

1 Definition

Project Overview

Inaccuracies in auto insurance company's accident claim prediction increase the cost of insurance for good drivers and reduce the price for bad ones. Therefore, predicting the probability that a driver will initiate an auto insurance claim can guide auto insurance companies during the decision making and pricing processes, such that they could come up with more fairer auto insurance plans for their customers. As machine learning techniques have been widely applied to this type of problem (e.g., Rondović et al., 2017), Porto Seguro, one of Brazil's largest auto and homeowner insurance companies, is interested in exploring new, more powerful machine learning methods. A more accurate prediction will allow them to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

The motivation is to predict if a driver would claim an accident given 57 features that describe several aspects of a driver. This project is from a Kaggle playground competition Kaggle Playground Competition. The objective is not only to obtain accurate prediction for accident claims, but also to learn advanced machine learning methods, particularly regression techniques.

Kaggle is a platform for data science competitions where the participants are challenged to build models to solve real-world machine learning problems. Kaggle usually provides a training dataset and a test dataset, and the task of the participants is to build a model based on the training dataset and make predictions on the test dataset. The participants can submit their predictions which will be evaluated by specific evaluation metrics. The competition has a leaderboard for the participants to compare their results with others. In this project, I will focus on studying novel machine learning techniques and making my score close to those top rankers.

Problem Statement

The goal of this project is to train a model based on the given data to predict the probability that an individual would file a car accident claim in the next year. The related tasks to this problem include:

- Exploratory data analysis – Understanding the type of features in the dataset (e.g., numeric, categorical, binary, etc.), how much missing data exist, and whether certain features are skewed, etc. These properties are crucial for constructing an accurate model in the end.
- Feature preprocessing – Preprocess the data using the observation from the exploratory data analysis, including but not limited to feature transformation, data type transformation, discarding certain features.
- Benchmark modeling – Creating a model using standard techniques for the problem in order to set up a benchmark for the future modeling improvement.
- Model improvement – Tuning model parameters to improve the model performance.

Metrics

The Normalized Gini Coefficient will be used to as the evaluation metrics for this project. The Gini coefficient measures the inequality among values of a frequency distribution. During calculation, actual observations are sorted from the largest to the smallest predictions. Predictions are only used for ordering actual observations; therefore, the relative magnitude of the predictions are not used during calculation. The scoring algorithm then compares the cumulative proportion of positive class observations to a theoretical uniform proportion. The Gini Coefficient ranges from approximately 0 for random guessing, to approximately 0.5 for a maximum score. The Normalized Gini Coefficient adjusts the score by the theoretical maximum, i.e., Gini(actual, predicted)/Gini(actual, actual), therefore, the maximum score is 1.

2 Analysis

Data Exploration

The dataset consists of numerical and categorical features as well as some pre-calculated ones. Specifically, there are nearly 600,000 observations, each with 57 features. The features cover many aspects of one customer and are believed to be sufficient to describe this individual's driving habit. The dataset was labeled by identifying whether the customer files an auto accident claim in the past year. It is found that 21694 out of 595.212 individuals claimed an accident, 3.64% of the dataset. In the dataset, Porto Seguro labeled features that belong to similar groupings in the feature names (e.g., ind, reg, car, calc). In addition, feature names include the post-fix bin and cat to represent binary and categorical features, respectively. Features without these designations are either continuous or ordinal. The dataset was split into two pieces: training (80%) and test sets (20%). Stratified k-fold cross-validation was used to ensure class balances across each subset. Our task is to predict the probability of a customer files a claim in the test dataset using the model trained from the training dataset.

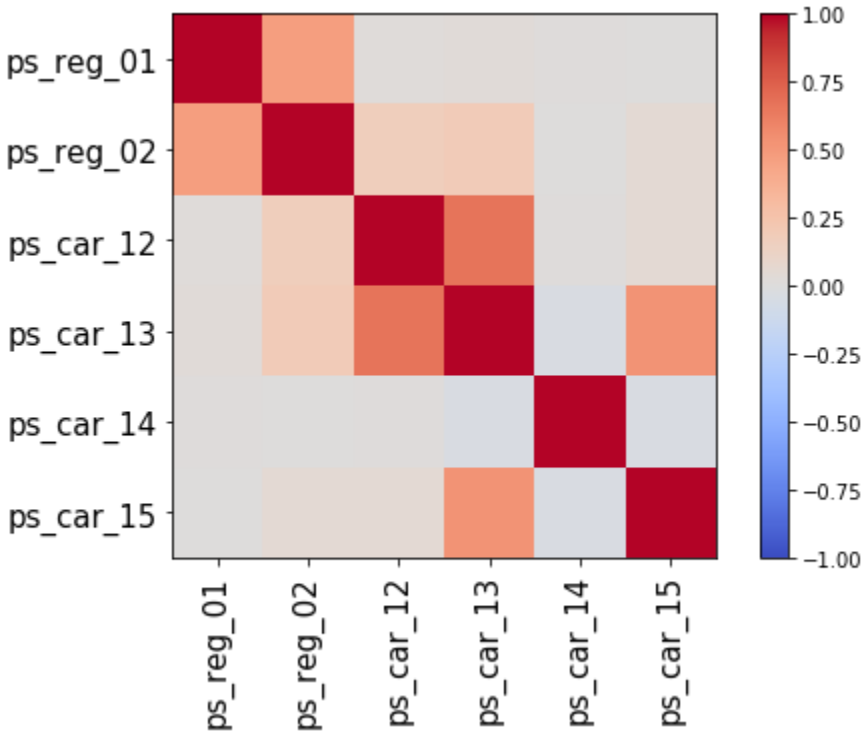


Figure 1: Correlation Matrix of Continuous Data Features.

Exploratory Visualization

Figure 1 shows the correlation matrix among the numerical features. There are several variables which are strongly correlated with each other, e.g., ps_reg_01 and ps_reg_02, ps_car_13 and ps_car_12, and ps_car_13 and ps_car_15.

Figure 2 shows a two-way table of the binary data features. As shown in the figure, most features have both their values labeled for both targets, i.e., an accident claim has been filed (1) or not (0).

Figure 3 shows a two-way table of the ordinal data features. As shown in the figure, most features have all their values labeled. No particular relationship between features and targets can be found in the data.

The categorical data features will not be shown here as they have patterns similar to the ordinal data features. To include all the categorical features into the machine learning model, the categorical variables will be transformed into some dummy variables via one-hot encoding, which takes values 0 or 1 to indicate the absence or presence of some categorical effects.

After the above-mentioned data-processing, the dataset contains 214 features in total and will be split into two pieces: 80% for the training dataset and 20% for the test dataset.

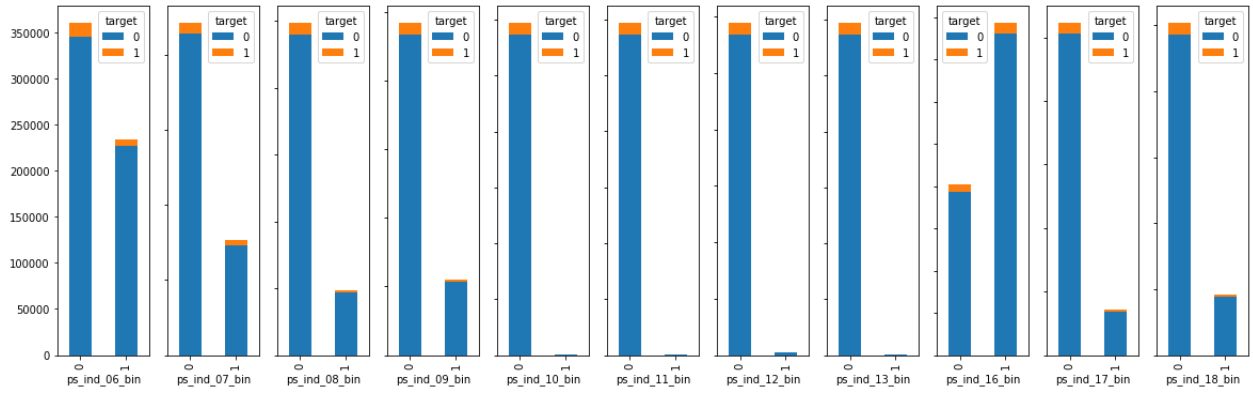


Figure 2: Two-Way Table of Binary Data Features. Target value 1 represents filed accident claims; target value denotes no claim.

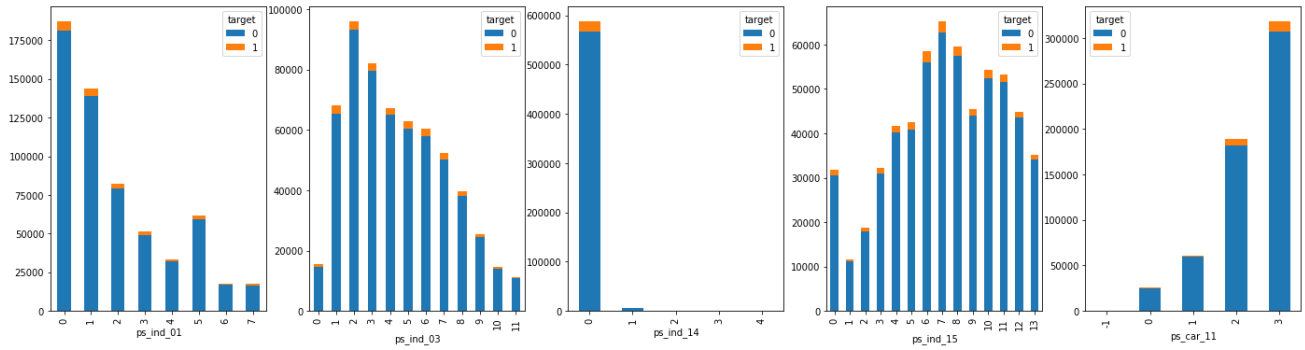


Figure 3: Two-Way Table of Ordinal Data Features. Target value 1 represents filed accident claims; target value denotes no claim.

Algorithms and Techniques

As the purpose of this project is to explore advanced regression techniques, the ensemble learning methods are to be implemented to achieve higher evaluation score. The ensemble learning methods use multiple learning algorithms to obtain better predictive performance than any single one of the learning algorithms. Common types of ensembles are Bagging, Boosting, Bayesian parameter averaging, Bucket of Models, etc. We will utilize a technique called stacking in this study. Stacking, or stacked generalization, was introduced by Wolpert (1992). The method has been widely implemented for various machine learning problems. One famous example is the top-performer solution of the Netflix Prize competition, Feature-Weighted Linear Stacking Sill et al. (2009).

The idea of stacking is to use a model or stacker to combine all the previous model predictions. Figure 4 shows an illustration of a 2 level 5 folds 5 models stacking approach. First, the training data is split into five folds, then we iterate over the five models. During each iteration, each base model is trained by four folds of the training data, and one prediction is made by the trained model using the remaining one fold of training data. In the meanwhile, each base model also makes a prediction using the whole test dataset. After iterating over the five folds, we will have the prediction of the whole training dataset from five base models and five predictions of the test

dataset. We then use the prediction of the training dataset as new features to train the second level model (stacker). Finally, we average the five predictions of the test dataset as input for the trained second level model, and give prediction for the test dataset.

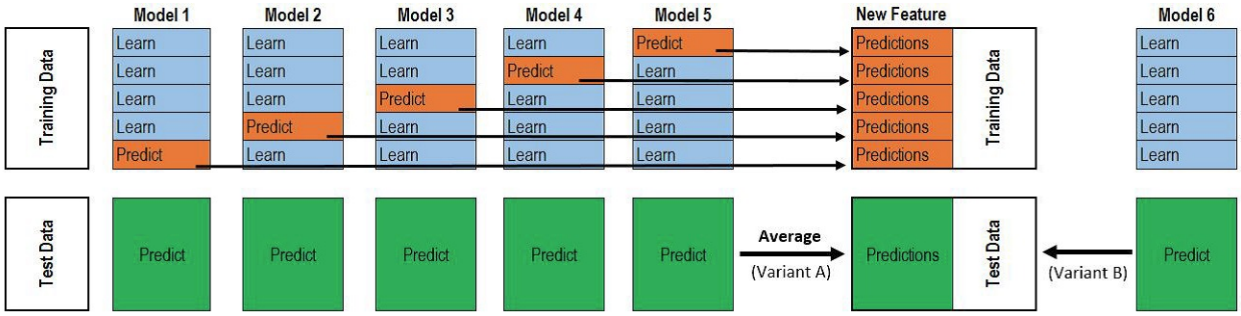


Figure 4: An Illustration of a Two-Level Five Folds Stacking. Courtesy of Wille.

Benchmark

We implement Gaussian Naive Bayes as our benchmark model for this project. The default parameters in the scikit-learn library's Native Bayes package will be used in the benchmark, which yields a Gini coefficient of 0.18559.

3 Methodology

The overall methodology and modeling approach can be summarized as data preprocessing, first level base model training, second model stacking, and prediction on test dataset. The details of this approach will be explained in the following.

Data Preprocessing

- Feature selection

First, we eliminate features with too much missing data. In the training dataset, features "ps_reg_03", "ps_car_03_cat", and "ps_car_05_cat" have over 15% of data missed, therefore will be discarded. Moreover, the pre-calculated features, i.e., those with "calc" in the feature names, will not be used in the machine learning as suggested by the Kaggle public kernel and justified by the 1st place solution of this project.

- Categorical features

The categorical features are processed by one-hot encoder which converts categorical variables into dummy or indicator variables.

Models	Normalized Gini Coefficient
Gaussian NB	0.18559
Decision Tree	0.02269
K Neighbors	0.04058
Linear SVM	0.00020
Random Forest	0.00185
Extra Tree	0.07383
Elastic Net	0.00020
LightGBM	0.27625

Table 1: Preliminary Model Scores.

Implementation

The first step is to choose the regressors as the first-level base models for the stacking method. For this purpose I tried Decision Tree regressor, K neighbors regressor, Linear Support and Vector Machine regressor, Random Forest regressor, Extra Trees regressor, Elastic Net regressor, and Gradient Boosting regressor. The Gradient Boosting regressor is implemented by the use of the LightGBM (2018) package. All the other regressors are from the scikit-learn library. All the regressors are used with their default parameter values without fine tuning. The preliminary test scores are shown in Table 1. It is found that probability estimates give higher Gini coefficient than regular class label prediction for this problem. Therefore, we use regular class label prediction from some regressors only if a probability estimate is not an output option. As the preliminary model performance shows, LightGBM provides the highest Gini coefficient. The simple Naive Bayes Model (Gaussian NB) gives a second highest score with model-default parameters. The other model, however, show much weaker performance.

The idea of stacking is based on using the predictions from the first level models as new features for the second level model, and to make prediction based on that. Two challenging parts exist in the implementation of stacking. The first one is to choose the base models for the first level prediction. The second one is to find the hyperparameters for both the first-level base models and the second-level stacker. As the LightGBM approach outperforms the other models in the preliminary tests, I will use it for the first-level model. LightGBM with different combinations of hyperparameters will be applied to construct the base models. Then I will adopt the logistic regression classifier as the second level model.

Refinement

Two approaches are conducted to refine the model behavior. At the first-level base models, the training data is split into several folds to train the base models. I used different fold and model numbers for the base models and default hyperparameters for the second-level stacker. First, I ran the model for 2 first-level models with fold number increasing from 2 to 5. Then, I tested the model for 5 folds with model number growing from 2 to 4. As Figure 5 shows, the model performance improves with fold and model numbers. Furthermore, the grid search technique is

conducted for the second-level stacker model. A few hyperparameters are tuned for the logistic regression classifier to achieve the highest Gini score.

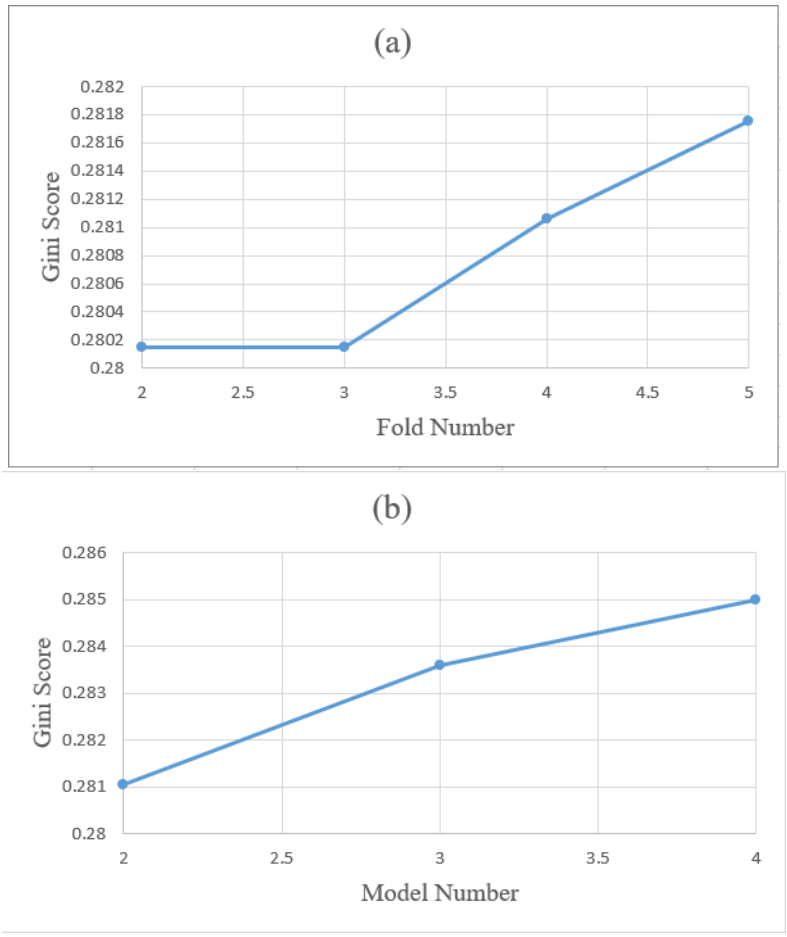


Figure 5: Gini Score as a Function of Fold Number (a) and Model Number (b).

4 Results

Model Evaluation and Validation

It has been found in the refinement that increasing the fold and base-model numbers can increase the prediction score. However, the score increases tiny (at the order of 0.001) when I increase the fold number from 2 to 5 and the base-model numbers from 2 to 4, although the computational time increases greatly. As the main project purpose is to learn about the ensemble stacking approach, only reasonable amount of computational time should be allowed. Therefore, the final fold and base-model numbers are choose to be 5 and 4, respectively. The increase in these numbers results in higher Gini coefficient from 0.28014 to 0.28499 (Figure 6). It is also worth it to mention that, the stacking approach is supposed to work better when the base models are as uncorrelated as possible. Some effort has been put to try different sklearn-kit regressors as components of the

first-level model. This approach, however, gives lower Gini scores. Therefore, we will only use LightGBM with different hyperparameters for the base models.

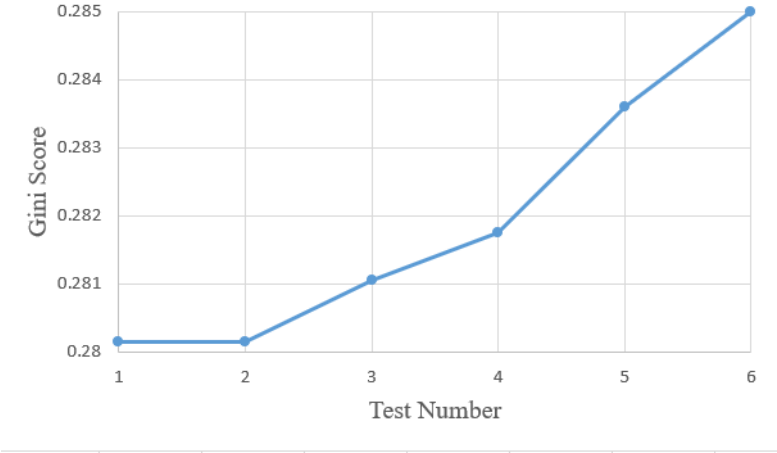


Figure 6: Model Performance improvement at different stage of model development.

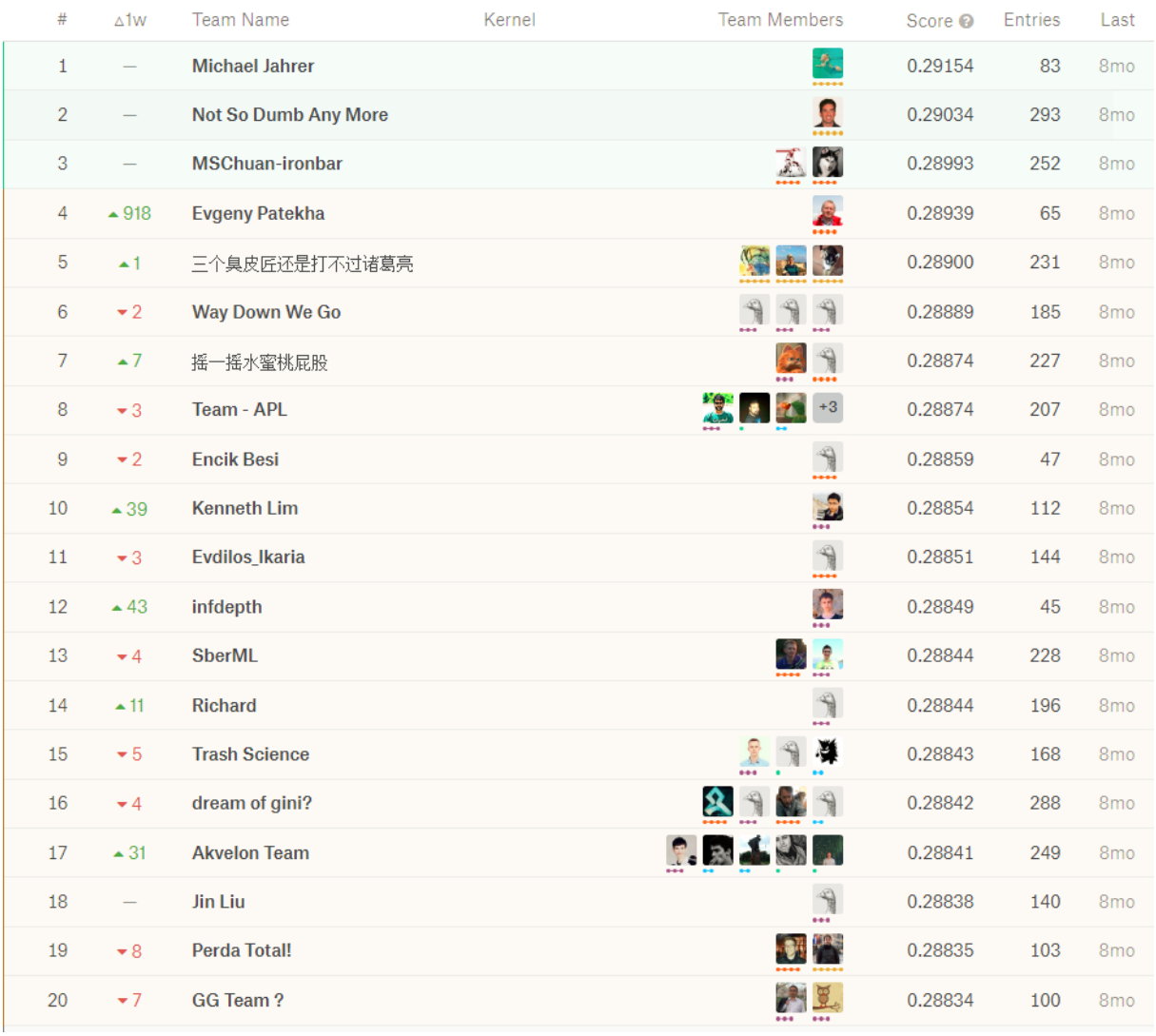
Justification

In summary, the stacking approach in this project greatly improves the Gini score from 0.18559 (Gaussian NB) to 0.28499 (stacked LightGBM). The result is encouraging and provide lots of things to think, learn and explore.

5 Conclusion

Free-Form Visualization

As Kaggle's Porto Seguro Safe Driver Prediction has been completed, my ultimate goal in this project is to obtain a score as close as possible without putting too much computational time. I will use the Kaggle Leaderboard top ranker's score as my final project visualization (Figure 7). My score can be further improved by increasing fold and model numbers, which is, however, not the interest of this project.



#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	Michael Jahrer			0.29154	83	8mo
2	—	Not So Dumb Any More			0.29034	293	8mo
3	—	MSChuan-ironbar			0.28993	252	8mo
4	▲ 918	Evgeny Patekha			0.28939	65	8mo
5	▲ 1	三个臭皮匠还是打不过诸葛亮			0.28900	231	8mo
6	▼ 2	Way Down We Go			0.28889	185	8mo
7	▲ 7	摇一摇水蜜桃屁股			0.28874	227	8mo
8	▼ 3	Team - APL		+3	0.28874	207	8mo
9	▼ 2	Encik Besi			0.28859	47	8mo
10	▲ 39	Kenneth Lim			0.28854	112	8mo
11	▼ 3	Evdilos_Ikaria			0.28851	144	8mo
12	▲ 43	infdepth			0.28849	45	8mo
13	▼ 4	SberML			0.28844	228	8mo
14	▲ 11	Richard			0.28844	196	8mo
15	▼ 5	Trash Science			0.28843	168	8mo
16	▼ 4	dream of gini?			0.28842	288	8mo
17	▲ 31	Akvelon Team			0.28841	249	8mo
18	—	Jin Liu			0.28838	140	8mo
19	▼ 8	Perda Total!			0.28835	103	8mo
20	▼ 7	GG Team ?			0.28834	100	8mo

Figure 7: Kaggle Public Leaderboard.

Reflection

The objective of this project is to predict car accident claims using advanced machine learning techniques. The dataset has over 600,000 observations with 57 features for each observation. An preliminary data analysis reveals that some features have too much missing data, therefore should be discarded. I conducted several data pre-processing steps to make the data ready for training. One of the difficult parts in data preprocessing is that I was including the pre-calculated features in training and obtaining bad results, until I found out that a lot of the Kaggle users were having the same issue. It turns out that dropping these features leads to smaller input data and higher prediction score. Then I trained several regressors with their default parameters. The LightGBM approach gives a much higher score than the rest of the benchmark test models, thus I used it to construct the first-level base model for the stacking approach. The other challenge is to choose suitable parameters for the two level of models. For the first-level model, I was using a set of

parameter as the benchmark, and making modifications based on that. The LightGBM models with different parameters form the base models. For the second-level stacker model, I was using a grid search approach for its parameters. The first-level takes a much longer time to train compared to the second-level one, therefore I wrote two separate modules for each model.

Improvement

This project gives several directions for future improvement. The first one regards data preprocessing which includes missing data replacement, advanced feature selection, and so on. The second one is to use more complicated machine learning techniques to construct the model input. For example, Kaggle's 1st-place top ranker used a Denoising Autoencoders (DAE) to process the input data and neural networks (thousands of neurons and three layers) to make the final prediction. He obtained a score of 0.29698, only 0.02 higher than a simple LightGBM approach without fine tuning for parameters. Obviously, there is a trade-off between the computational resource and the actual improvement in the score.

References

Biljana, Ljiljana Kašćelan, Vujica Lazović, Tamara Đuričković, 2017. A nonparametric data mining approach for risk prediction in car insurance: a case study from the Montenegrin market, Information Technology for Development 0:0, pages 1-30, https://doi.org/10.1080/1331677X.2016.1175729.

LightGBM: http://lightgbm.readthedocs.io/en/latest/index.html.

Wolpert, D. H. (1992). Stacked generalization. Neural networks, 5(2), 241-259.

Joseph Sill, Gabor Takacs, Lester Mackey, David Lin, Feature-Weighted Linear Stacking, https://arxiv.org/abs/0911.0460.