Travelers Insurance Fraud Prediction

King Yiu Suen (Group 2)

Abstract

This paper provides a brief report on how my group approached the Travelers insurance fraud prediction competition, including data cleaning, feature engineering, model selection, and finding important variables. Business insights from the results, directions for improvement and a personal reflection of what I have learned from this competition were also provided. A weighted average of XGBoost and LightGBM was found to have the best AUC score.

1 Introduction

Insurance frauds are costly to insurance companies. In this Kaggle competition, we were challenged to predict which first party physical damage auto insurance claim was a fraud. A training dataset containing 17,998 rows and a test dataset containing 12,001 rows were given. Each row corresponds to a claim, uniquely identified by a claim number. The datasets had 23 predictors, including the information of the claimer, the characteristics of the damaged vehicle, and some details about the accident. The fraud indicator in the test dataset was NA. Our task was to build a model using the training dataset to predict the fraud indicator for each claim in the test dataset.

The remainder of the paper is organized as follows. In Section 2, we demonstrate how we cleaned the data (e.g., dealing with abnormal values and missing values). Section 3 present our feature engineering process (e.g., what new features we added and what existing features we did not use). Section 4 discusses what models we considered and how we selected our final model. Section 5 explains how I decided the classification threshold. Section 6 discusses which variables are important for interpretation. Section 7 provides the directions for improvement and a personal reflection on what I have learned from this competition.

2 Data cleaning

There were 162 missing values in four different variables. Mean and mode imputations were implemented for missing values on continuous and categorical variables respectively. We assumed that the number of missing values relative to the sample size was so small that the imputation method would not make a significant difference. Abnormal values such as `age_of_driver > 100` or `annual_income = -1` were also replaced by the mean of their respective variable. Three observations that were found to have `fraud = -1` were removed. All categorical variables were transformed into dummy variables, because some of the models we tried only accepted numerical values.

3 Feature engineering

Some of the existing features in the dataset were manipulated so that they could be more informative. For instance, we broke the claim date into year, month, day of month, and day of week. The zip codes were replaced by their corresponding state, longitude and latitude. We also created a new feature named rating per claim, which was a transformation of two existing features: `safty_rating / (past_num_of_claims + 1)`. Our rationale was that the safety rating and the past number of claims are both indicators of a person's credibility but the credibility indicated by safety rating can be "consumed" by the past number of claims. For a person with a high safety rating, the chance of fraud is still high if he/she has a high number of past claims. The addition of one in the denominator was to avoid dividing by zero. The safety rating and rating per claim are plotted against the fraud rate in Figures 1 and 2. The variance of fraud rate given rating per claim is much smaller than the variance of fraud rate given safety rating on the left tail, suggesting the potential usefulness of this new feature.

In addition to the existing features, we added some features that might have predictive values but were not included in the original dataset. First, we hypothesized that people are more motivated to commit fraud when the economy is bad. Thus, we found three indices that reflect the macroeconomic situation on the claim date: interest rate, Standard & Poor's 500 (S&P 500) and the unemployment rate at the state level. Moreover, we postulated that people who intend to commit fraud are more unwilling to disclose their personal information. Hence, the number of missing values for each observation was computed and was used as a predictor.

Not all features were used as the predictors of our final model, as we believed that some of them were simply introducing noise to our model and exposed us to the risk of over-fitting. Therefore, a preliminary study was conducted to examine which feature had clearly no predictive value. Features were removed if they were consistently at the bottom of the feature importance list in Adaboost (Rätsch, Onoda, & Müller, 2001), random forest (Breiman, 2001), XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017), and had zero coefficients in Lasso (Tibshirani, 1996). They included: the vehicle color, all features related to the claim date, S&P 500 index, unemployment rate and the state of the zip code.

4 Model selection

The models that we mainly considered were the models mentioned in the previous section: Lasso, random forest, Adaboost, XGBoost and LightGBM. We used a greedy approach to search for the optimal set of parameters manually. For each model, we started with the set of default parameters $\boldsymbol{\theta} = \{\theta_1^{(0)}, ..., \theta_M^{(0)}\}$. For $m = 1, ..., M$, we used grid search to find the optimal value of the m-th parameter given $\boldsymbol{\theta} \setminus \theta_m^{(0)}$, call it θ_m^*; then, update $\theta_m^{(0)}$ with θ_m^* in $\boldsymbol{\theta}$. Finally, we repeated the same process with $\{\theta_1^*, ..., \theta_M^*\}$ as the initial set and used the resulting set of parameters $\boldsymbol{\theta}$ for model comparison. We also tried Bayesian optimization for parameter tuning, but the cross-validation scores seemed to be lower than those obtained from manual search, possibly because we did not run enough iterations or we did not know how to use it properly.

The performance of the optimized models were then evaluated using a 5-fold cross valida-

tion. Their AUC scores were displayed in Table 1. LightGBM performed the best, followed by XGBoost. We then considered taking the weighted average of the two best models, as one of the models may be good at predicting part of the observations, and the other model may be good at predicting the other part. By combining two models, we speculated that the performance would be better. Different sets of weights were assessed again by a 5-fold cross validation. The optimal weights we found were 0.6 and 0.4, which achieved an AUC score of 0.73013 locally, the highest among all models we tried. Thus, we used this as our final model. It achieved an AUC score of 0.74961 in the public leaderboard (1st place) and an 0.75462 in the private leaderboard (2nd place).

The model fitting and evaluation was all performed in Python. The code for fitting our final model was presented in Appendix B.1.

5 Prediction

After obtaining the predicted probabilities of every observation, $\hat{p}_i$, from the final model, we need to choose a threshold c such that observation i is classified as fraud if $\hat{p}_i \geq c$. Following the instructions, a false negative error needs to be five times more costly than a false positive error. Formally, the cost of observation i under the use of threshold c is

$$\delta_{i|c} = \begin{cases} 5 & \text{if } y_i = 1 \text{ and } \hat{y}_i = 0 \\ 1 & \text{if } y_i = 0 \text{ and } \hat{y}_i = 1 \\ 0 & \text{if } y_i = \hat{y}_i \end{cases} \tag{1}$$

where y_i and $\hat{y}_i$ are the true and predicted labels for observation i respectively.

The `optimize` function in R was used to find the value of c that minimized the total misclassification cost of the training dataset:

$$\arg\min_{c} \sum_{i \in S_{\text{train}}} \delta_{i|c} \tag{2}$$

where S_{train} the set that includes all the observations of the training dataset. The optimal threshold c^* was found to be 0.3885591.

To estimate the total cost of our predictions on the test dataset, the expected cost of a single prediction in the training set was multiplied by the number of observations in the test dataset:

$$\left(\frac{1}{n_{\text{train}}} \sum_{i \in S_{\text{train}}} \delta_{i|c^*} \right) \cdot n_{\text{test}} \tag{3}$$

where n_{train} and n_{test} are the numbers of observations in the training and test datasets respectively. The estimated cost was found to be 6226.421. The R code implementation is presented in Appendix B.2.

6 Variable importance

To investigate which variable is most important for the underlying data generation process, we can examine the feature importance values generated by the models mentioned above.

However, the variables selected by one method alone could be unstable, and there is no obvious way to combine the feature importance values from these models, as they are measured in different scales.

In this study, Sparsity Oriented Importance Learning (SOIL; Ye, Yang, & Yang, 2018) was used. SOIL is able to combine the solution paths of multiple high-dimensional variable selection methods, thereby providing more reliable importance values than a single method. The union of solution paths of Lasso (Tibshirani, 1996), Adaptive Lasso (Zou, 2006), SCAD (Fan & Li, 2001), and MCP (Zhang et al., 2010) was used here. Based on a preliminary study, each of them had a decent AUC score (≥ 0.7). To screen out unimportant variables, only variables whose importance value is larger than a certain threshold should be kept. The range of the variable importance in SOIL is $[0, 1]$. To balance the cost of over-fitting and under-fitting, the threshold was set to 0.5. This procedure was implemented in R and the code is presented in Appendix B.3.

A total of 17 features was found to be important by SOIL. Table 2 shows the importance value of each feature. Whether or not the driver changed living address in past 1 year was important, possibly because the driver would like to cover their tracks if they are planning to commit fraud, or maybe it is a sign that the driver is financially unstable - they have to move around a lot. When people are more financially stable such as having higher annual income and owning an apartment, they are less likely to commit fraud. The age of driver was found to be important as well. This may be due to the fact that the age is strongly correlated with the annual income ($r = 0.98$). Unsurprisingly, the education background, the past number of claims and whether a witness is present affect the reliability of the claim. Fraud is also more probable if the channel of policy purchasing is online, possibly because people feel more comfortable to lie when they are not facing a real person. As we hypothesized, higher interest rate indicates bad economy, which then leads to more motivation to commit fraud. Higher estimated claim payout is also associated with a higher probability of fraud. It is possible that people do not even bother to fraudulate an accident if the reward is too small compared to the risk of being caught.

The probability of fraud is higher when the accident occurs in highway. However, it is possibly because accidents on the highway are just much more visible, so it is easier to find out if the accidents are fraudulent. When the accident is reported to occur in a local neighborhood or a parking lot, there may not be any surveillance cameras or witnesses; hence, some of the non-fraudulent cases may actually be fraudulent, but they are too difficult to investigate. The year of claim was also found important as the fraud rate in 2016 (17.1%) is higher than 2015 (14.2%). But since the range of this variable is too small (all the observations in the training set are in 2015 and 2016), it is difficult to generalize the interpretation that the more recent the year, the higher the probability of fraud.

Marital status, gender, longitude, vehicle color and vehicle category are also related to fraud but for unclear reasons.

7 Discussions and Conclusions

7.1 Directions for improvement

Some of the new features we added did not seem to be useful, for example, S&P 500 index and unemployment rate. It is very possible that the period we observed was too short.

Since an economic cycle may last for more than five years, the fluctuation of the economy in merely two years (2015 and 2016) may not be significant enough to make a difference. More data need to be collected over a longer period of time to verify whether the macroeconomic situation is an important variable.

There are some potentially useful features that are not included the dataset. First, insurance claim denial history. If a driver has been denied before, regardless of the type of insurance, this should raise a red flag. Second, crime rate. As suggested by SOIL, longitude is an important variable, which indicates that geographic information may be useful; however, the reason is unclear. A potentially more informative feature related to the geography is the crime rate of the neighborhood. Third, social media profile. If a driver has had a horrible car accident, they should be busy putting their life back together, instead of updating their social media profile and showing off their lifestyle.

7.2 Personal reflection

Through the competition and the presentation of other groups, I learned about gradient boosting methods, such as CatBoost, XGBoost and LightGBM, and that they tend to have better prediction accuracy than statistical methods we typically learn in the classroom, for example, logistic regression and linear discriminant analysis, as none of the groups used these textbook methods as their final model. Gradient boosting methods will definitely be the first methods I try if I participate in a data science competition again. I also learned that a single model alone is not enough to provide the best prediction. Most groups, including my own group, used some kind of model averaging in their final model, either using a weighted average or stacking. One obvious drawback is that interpretation is difficult. When it comes to variable importance, all the groups only showed the feature importance plot given by each one of the models separately, which could not give a coherent picture.

Moreover, I have seen how creative we can be in feature engineering. From Group 3, I learned that principal component analysis (PCA) can be applied in this context. When the age and the annual income are highly correlated, they used PCA to reduce the dimensionality of the data while retaining as much variation as possible, by only keeping the first principal component. I have also seen many ways to generate new and meaningful features based on the existing features, both from my own group and other groups, such as converting zip codes to longitude and latitude, and doing transformation of existing variables. As a psychology student, this is new to me. I used to think that the only predictors we can use are the variables we manipulate in our experiment, but this competition has shown me that we can create our own feature to improve the prediction accuracy.

References

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, *96*(456), 1348–1360.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems* (pp. 3146–3154).

Rätsch, G., Onoda, T., & Müller, K.-R. (2001). Soft margins for adaboost. *Machine learning*, *42*(3), 287–320.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Ye, C., Yang, Y., & Yang, Y. (2018). Sparsity oriented importance learning for high-dimensional linear regression. *Journal of the American Statistical Association*, 1–16.

Zhang, C.-H., et al. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, *38*(2), 894–942.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, *101*(476), 1418–1429.

Appendix A

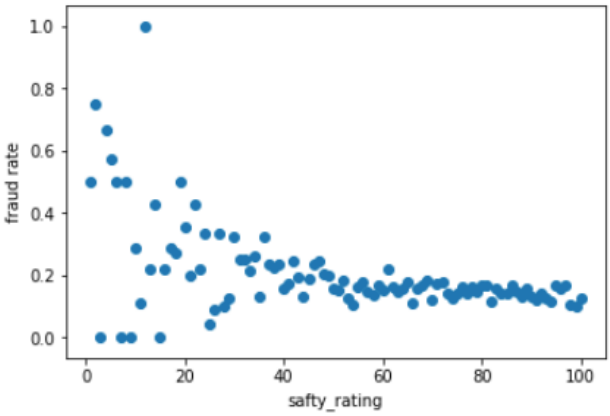

Figure 1. Scatter plot between safety rating and fraud rate

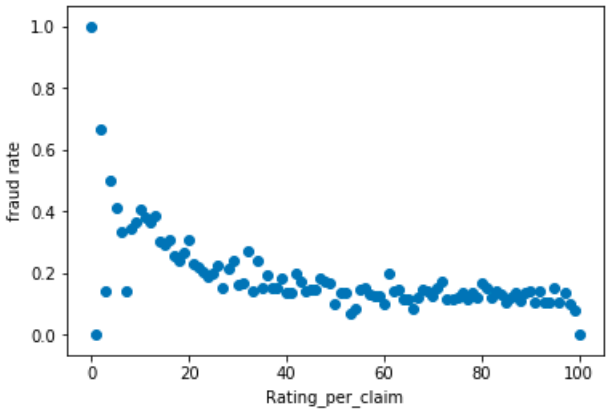

Figure 2. Scatter plot between rating per claim and fraud rate

Table 1. AUC scores of Lasso, random forest, Adaboost, XGBoost, and LightGBM under a 5-fold cross validation

Model	AUC
Lasso	0.70910
Random forest	0.71169
Adaboost	0.71996
XGBoost	0.72539
LightGBM	0.72920

Table 2. Variable importance measured by SOIL

Feature	Importance
`marital_status`	1.000
`high_education_ind`	1.000
`address_change_ind`	1.000
`past_num_of_claims`	1.000
`witness_present_ind`	1.000
`accident_site` (parking lot)	1.000
`gender` (M)	1.000
`age_of_vehicle`	1.000
`age_of_driver`	1.000
`interest_rate`	1.000
`safty_rating`	0.9995
`longitude`	0.9861
`accident_site` (local)	0.9154
`channel_online`	0.7606
`living_status` (own)	0.9498
`claim_est_payout`	0.9004
`vehicle_category` (large)	0.8966
`annual_income`	0.8608

Appendix B.1 Python Code for model fitting

```
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import VotingClassifier

lgbm_params = {
        "boosting_type": "gbdt",
        "objective": "binary",
        "num_boost_round": 800,
        "feature_fraction": .321,
        "bagging_fraction": 0.50,
        "min_child_samples": 100,
        "min_child_weigh": 35,
        "max_depth": 3,
        "num_leaves": 2,
        "learing_rate": 0.15,
        "reg_alpha": 5,
        "reg_lambda": 1.1,
        "metric": "auc",
        "max_bin": 52,
        "colsample_bytree": 0.9,
        "subsample": 0.8,
        "is_unbalance": "true"
}

xgb_params = {
        "max_depth": 3,
        "learning_rate": 0.05,
        "n_estimators": 160,
        "objective": "binary:logistic",
        "gamma": 0.3,
        "min_child_weight": 5,
        "max_delta_step": 0,
        "subsample": 0.8,
        "colsample_bytree": 0.785,
        "colsample_bylevel": 1,
        "reg_alpha": 0.01,
        "reg_lambda": 1,
        "scale_pos_weight": 1,
        "seed": 1440,
        "missing": None
}

lgbm = LGBMClassifier(**lgbm_params)
xgb = XGBClassifier(**xgb_params)
```

```
ma_sub = VotingClassifier(
        estimators = list(zip(["lgbm", "xgb"], [lgbm, xgb])),
        voting = "soft", weights = [6, 4]
)

ma_sub.fit(X_train, y_train)
y_preds = ma_sub.predict_proba(X_test)
```

Appendix B.2 R Code for prediction

```
library(rBayesianOptimization)
library(dplyr)

train <- read.csv("prob_train.csv")
test <- read.csv("prob_test.csv")
n_train <- nrow(train)
n_test <- nrow(test)

# compute the cost of errors
calculate_cost <- function(df, threshold) {
        df <- df %>%
                mutate(decision = ifelse(prob >= threshold, 1, 0)) %>%
                filter(decision != fraud) %>%
                mutate(cost = ifelse(fraud == 1 & decision == 0, 5, 1))
        return(sum(df$cost))
}

# find the optimal threshold
objective_function <- function(threshold) {
        return(calculate_cost(train, threshold))
}
optimal <- optimize(objective_function, c(0, 1))
threshold <- optimal$minimum

# apply the optimal threshold to the
# predicted probabilities of the test dataset
claim_number <- read.csv("umn_comp_2018_test.csv") %>%
        select(claim_number)
prediction <- test %>%
        transmute(fraud = ifelse(prob >= threshold, 1, 0)) %>%
        bind_cols(claim_number, .)
write.csv(prediction, "prediction.csv", row.names = FALSE)

# estimate the cost of errors on the training dataset
cost_train <- optimal$objective / n_train

# estimate the cost of errors on the test dataset
cost_test <- cost_train / n_train * n_test
```

Appendix B.3 R Code for SOIL

```
library(SOIL)
library(dplyr)

df <- read.csv("train_data.csv")
X <- as.matrix(df %>% select(-fraud))
y <- as.matrix(df %>% select(fraud))
n <- nrow(df)

soil <- SOIL(X, y, n_train = ceiling(n / 2), no_rep = 100,
        n_train_bound = ceiling(n / 2) - 2, n_bound = n - 2,
        psi = 1, family = "binomial", method = "union",
        weight_type = "ARM", prior = TRUE)

data.frame(colnames(soil$importance), t(soil$importance)) %>%
        rename(feature = colnames.soil.importance.,
               importance = t.soil.importance.) %>%
        filter(importance >= 0.5) %>%
        arrange(desc(importance)) %>%
        mutate(importance = round(importance, 4))
```