Fast or furious? - User analysis of SF Express Inc

Gege Wen@gegewen, Yiyuan Zhang@yiyuan12, Kezhen Zhao@zkz

I. Motivation

The motivation of this project is to predict the likelihood for a S.F. Express user to file complaint for a certain delivery using machine learning algorithms. With the prediction, we will also determine the top triggers that make a user furious, therefore improve the users experience during a delivery.

The S.F. express company is the second largest carrier in China and we will focus on the company's domestic delivery business. This project is proposed based on the company's current business needs.

II. Related work

Through literature review, we found several papers related to our topic. The most relevant paper builds a system that helps automatically predict a patient's chief compliant based on vitals and nurses' description of states at arrival[1]. A linear support vector machine (SVM) on their selected features and achieved a F1 score of 0.913 on their perceptron algorithm. Their work inspires us on the choices of learning algorithms and feature simplification process. However, there is some differences in our problem since we need to select variables on bases of limited pre-existing features and we expect high recall rate due to industry needs.

We applied different models that were proved to be highly efficient in past literature: Wiener's paper[2] gives us a guideline to perform Random Forest classification. Chen[3] summarized XGBoost, which can help us get better results on a tree boosting system. Friedman[4] experimented on Stochastic Gradient Boosted Decision Trees(GBDT) and we also referred to their work on boosting process. For Feature selection, we applied several types of Least Absolute Shrinkage and Selection Operator(LASSO) mentioned in Tibshirani's[5], Chen's[6] and Zhang's[7] papers.

III. Dataset and Features

A. Dataset

The dataset comes from S.F.'s real delivery history, containing the industrial bar gun information including descriptions on the packages, senders, and receivers. The dataset represented in CSV format with 40 columns as attributes and 22000 rows as input entries. After cleaning null and irrelevant attributes, 27 attributes are remained as potential features.

Since delivery entries with complaint rarely occur comparing to all delivery histories, the dataset is highly unbalanced with 2000 complaint entries and 20000 non-complaint entries. To solve this issue, we created three balanced datasets each containing all 2000 complaint entries and 2000 randomly selected non-complaint entries. With in each balanced set, we split 75% to be the training set and 25% to be the test set. Note that a development set is also created within each training set, which contains 20% of the training set entries. The following feature and model selection process are performed on all three balanced datasets. Therefore, we can cross validate using the three datasets to achieve a optimal generalization during parameter tuning. To test the performance of the algorithms under real world condition, we also created a unbalanced test set where the complaint and non-complaint entries are in the ratio of 1:10.

B. Data Pre-processing

The features in our dataset exist in numerical, binary, and categorical formats. To combine all variables, we normalized the features using dummy variables into the form of a feature matrix. An example of this normalization is shown as follows:

$$\begin{cases} x_1 \to numerical & \in \mathbb{R} \\ x_2 \to categorical & \in \{A, B, C\} \\ x_3 \to binary & \in 0, 1 \end{cases}$$

If we have $x_1 = 5, x_2 = A, x_3 = 1$, our feature matrix will be:

$$\begin{bmatrix} x_1 \\ x_2B \\ x_2C \\ x_3 \end{bmatrix} \to \begin{bmatrix} 5 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Note that to avoid dummy variable trap, a categorical variable with n categories is projected to $n-1$ columns. The dimension of the resulting feature matrix is 87.

C. Feature Selection

We performed feature selection since our mapped feature matrix is very high dimensional. Using the 87 features, we feed our set 1,2 and 3 into a simple logistic regression and yield an average AUC of 0.53. Logistic regression with lasso has even worse performance with an average AUC of 0.52. Therefore, we used Z-test, lasso, recursive and stability selection to reduce the dimension of feature.

1) Z-Test Selection: We assumed that a feature is considered relevant to the predication when we reject the hypothesis that a coefficient equals to zero. When $Z > 2$, we have 95% confidence to reject null hypothesis and claim the coefficient is statistically significant.

2) Lasso method: We performed a cross validation to estimate the expected generalization error for each λ. The value of λ was chosen based on CVMSE to be 0.02. An example of λ verses CVMSE on set 1 is presented as Figure 1.

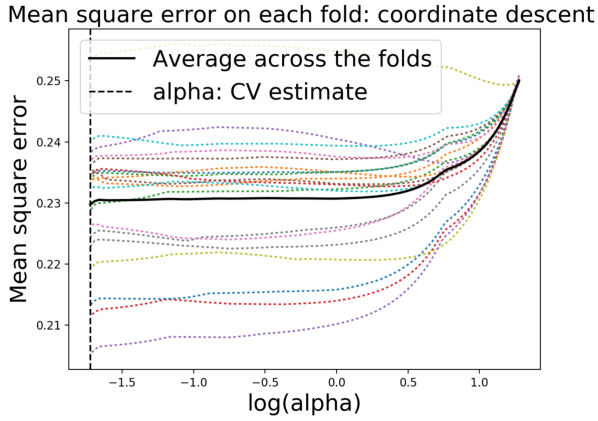

Fig. 1: λ vs. CVMSE

3) Recursive Selection: we eliminated least important features using the recursive method and compared the remaining feature in each dataset.

4) Stability Selection: using a randomized-lasso as our selection algorithm, we performed stability selection on three sets and eliminated the features that scored lower than 0.85 in all three sets.

Comparing the results from the four feature selection methods, we eliminated 40 features that were selected as insignificant by all methods. Also, 16 features were considered significant as shown in Figure 2 after combining the results from the four selection methods. The industrial meaning of these features can be further explored by the company in order to improve costumer experience.

dest_hq_code_3	consign_last6mon_cnt	cons_type1_2	send_la_6mon_comp_waynm
all_fee_rmb	dest_type_2	cons_type1_5	industry_type_level1_8
freight_rmb	dest_hq_code_3	waybill_type_dest_3	receiv_la_6mon_comp_waynm
cons_value	dest_hq_code_8	custtype_3	consigin_emp_hire_mon

Fig. 2: Significant features

IV. Model Selection

A. Baseline Model

We choose logistic regression as our baseline model. With the selected features and balanced data set, logistic regression model achieved an average AUC score of 0.64. Logistic regression with Lasso reach an average AUC of 0.65.

B. Tree-structured Model

In this real life problem, most features do not have a linear property. A tree structured model performs better at capturing the non-linearity. Therefore, to achieve better performance, we attempted Random Forest (RF), Gradient Boosting Decision Tree (GBDT) and XGBoost.

The RF algorithm take average on a number of parallel decision trees each has low bias and high variance. This algorithm will effectively control over-fitting while improve accuracy especially for high dimensional classification problem.

The GBDT algorithm builds a function which evaluates the loss of our method. Then we optimize it by making repetitive changes in the gradient direction of our loss function. Iterative improvement is made to reduce the cost in this algorithm.

The XGBoost is also a boosting algorithm but with L1 and L2 regularization as well as uses both first and second order gradient[3]. This method prune on user defined full trees to achieve better accuracy.

C. Experiments

For each of the three models, we performed a parameter selection process by using all the three

balanced datasets. For each set, we first apply our modeling method with default parameters and obtain our 'baseline' AUC score and accuracy on its corresponding development set. Then we partially tune up parameters to see if we are on the right track by checking the AUC score. After we finish tuning all the parameters, we achieve the 'best parameters' for this set.

Because our datasets are randomly sampled, the three sets perform differently. All algorithms gives relatively low AUC score on Set 2 and better score on Set 3. Therefore, to generalize the parameters and determine the best parameters for all dataset, we combine results from these three sets and build a combined parameter pool. Finally we selected parameters that can provide the best average performance on three sets and further check their results on test sets. A example of this process applied on RF is shown as follows.

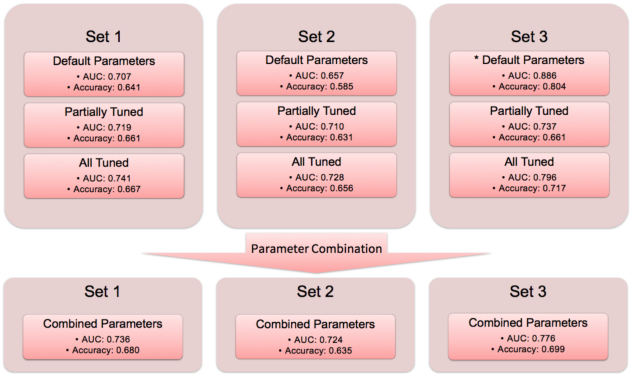

Fig. 3: Parameter selection process

Within each process, a parameter optimization technique is applied where we defined parameters to be either 'fixed' or 'inter-correlated'. A fixed parameters (e.g. max feature in RF) affects the AUC score independently. Therefore, the optimized value for this parameter is searched by by simply adjusting this parameter until reaching maximum AUC. However, inter-correlate parameters (e.g. depth, min sample split, and min sample leaf RF) affects the AUC result as a group. For these parameters, we make adjustment in a recursive manner where a portion of this parameters group is adjusted while the rest are fixed. We repeat this adjustment until maximum AUC is reached.

V. Results

A. Parameters

Using the above model selection methods, our final parameters for RF, GBDT and XGBoost are listed in Figure 4. Each method contain two sets of parameters, which are applied to balanced and unbalanced dataset respectively.

RF – Balanced / Unbalanced		
n_estimators = 110 / 300	max_depth = 17 / 20	min_samples_split = 50 / 60
min_samples_leaf = 1 / 1	max_features=10 / 10	
GBDT – Balanced / Unbalanced		
n_estimators = 105 / 115	max_depth = 11 / 11	min_samples_split = 60 / 60
min_samples_leaf = 6 / 6	max_features = 7 / 7	
XGBoosting – Balanced / Unbalanced		
n_estimators = 700 / 900	learning_rate = 0.07 / 0.07	gamma = 0.2 / 0.2
subsample = 0.75 / 0.8	colsample_bytree = 0.75 / 0.75	reg_lambda = 1e-5 / 1e-5
reg_alpha = 1e-5 / 1e-5	max_depth = 3 / 3	min_child_weight = 1 / 1

Fig. 4: Final parameters for the three models with balanced and unbalanced dataset

B. AUC

The AUC results from a new balanced set and a new unbalanced set are summarized in Figure 5. This results suggests that the GBDT and XGBoost model both give a relatively good AUC performance before tunning on unbalanced and balanced test set. After parameter tuning, GBDT performs the best on balanced set while XGBoost performs the best on unbalanced set.

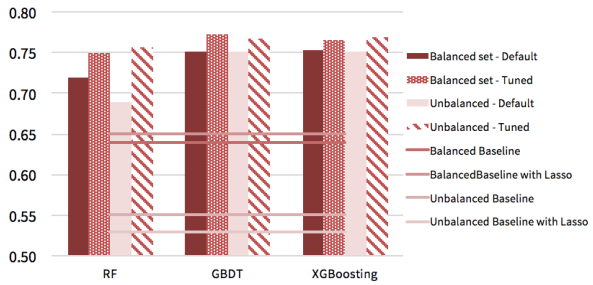

Fig. 5: Parameter selection process

VI. Discussion

A. Balanced Test set

Figure 6 shows the ROC and PR curve comparison when the algorithms are applied to the balanced test set. Comparing the ROC curve before and after parameter tuning, we can see a increase in AUC, which were represented by the area under the ROC curve due to tuning. RF, GBDT and XGBoost

benefited from the parameter tuning while the most significant improvement is observed on RF.

On the PR curve, we observed that the rate of precision quickly decreases when we attempt to improve the recall percentage of class-1, especially for the RF algorithm. After parameter tuning, we were able to smooth this decrease and improve the precision for a certain recall rate. The optimal precision is achieved by different algorithms under different desired recall rate.

In conclusion, for a balanced test set, we think that RF, GBDT and XGBoost all provide a good recall and precision. Since the test set is balanced, we can just use a threshold of 0.5 for all algorithms.

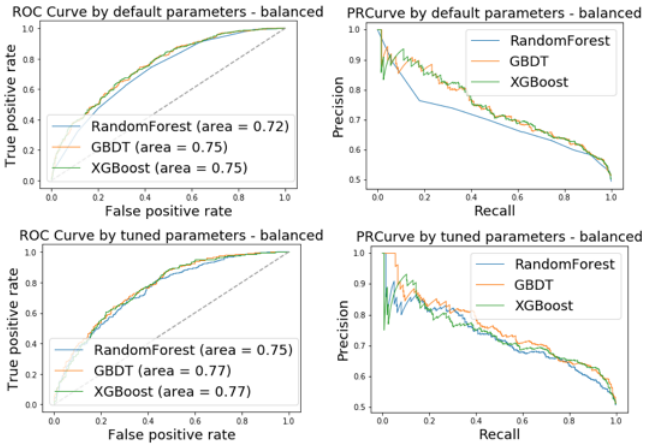

Fig. 6: ROC and PR Curve for class-1 before and after tuning on balanced dataset

We have also summarized the performance metrics of precision, recall and F1-score for both class-1 and class-0 predications as shown Figure 7. Since the dataset is balanced, we use F1 score here, which is the harmonic average of precision and recall. The performance metric indicates that GBDT and XGBoost out perform RF in terms of F1 score because these two algorithm gives better prediction on recall.

RF	Precision	Recall	F1-score
Balanced - 0 - Default/Tuned	0.65/0.69	0.69/0.67	0.67/0.68
Balanced - 1 - Default/Tuned	0.65/0.67	0.62/0.67	0.64/0.68
GBDT	Precision	Recall	F1-score
Balanced - 0 - Default/Tuned	0.69/0.70	0.65/0.68	0.67/0.69
Balanced - 1 - Default/Tuned	0.66/0.69	0.70/0.70	0.68/0.69
XGBoost	Precision	Recall	F1-score
Balanced - 0 - Default/Tuned	0.69/0.69	0.63/0.68	0.66/0.69
Balanced - 1 - Default/Tuned	0.66/0.67	0.71/0.70	0.68/0.69

Fig. 7: Performance Metrics on balanced dataset

B. *Unbalanced Test Set*

Figure 8 shows the ROC and PR curve comparison when the algorithms are applied to the unbalanced test set. From the ROC curves, we also observed that AUC is improved after we tune up all the parameters.

Comparing to the balanced dataset, precision rate decreases dramatically when we slightly improve on the percentage of recall from 0 to 0.2. This can be accounted to the class bias problem in the unbalanced data set. Due the overwhelming proportion of class-0 data (people who don't file a complaint is dominating in real-world dataset), our model tends to overfit to class-0. Therefore, when we make predictions on class-1, our fitted model will mistakenly classify a large portion of ones to zeros.

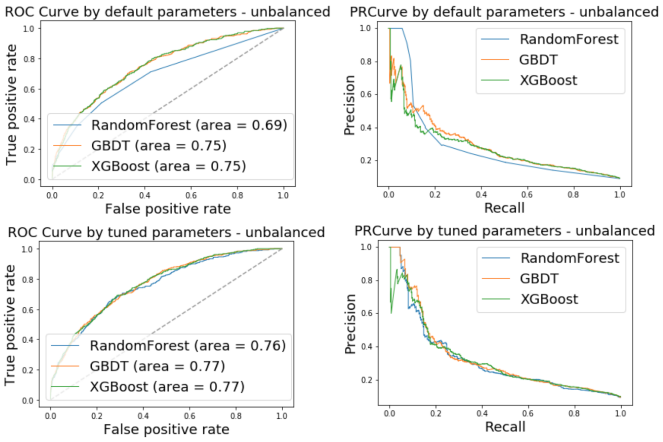

Fig. 8: ROC and PR Curve before and after tuning on unbalanced dataset

Companies' real needs is to identify the potential complaint customers. So we mainly focus on correctly predicting ones. To achieve this goal, we need to adjust β in F1-score function to increase statistical weight on recall rate. Here follow the principle that F-score is the harmonic average of precision and recall, and β is the weight on recall rate according to Van Rijsbergen's effectiveness measure. Considering all the ideas above and the fact that the component ratio for ones and zeros in our unbalanced dataset is 1:10, we eventually set $\beta = 3$.

Figure 9 illustrates how the recall rate and F_β score change as we modify the threshold of our model. Recall rate decrease dramatically if we raise the threshold value from 0 to 0.2. When the

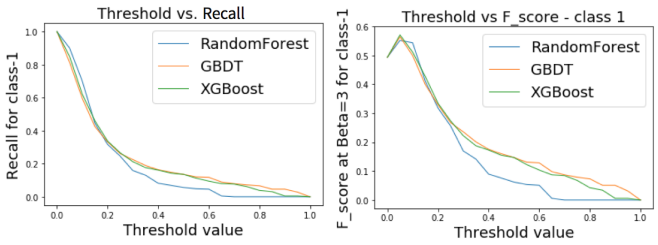

Fig. 9: Performance Metrics on unbalanced dataset

threshold is greater than 0.2, then the curve will decreases slowly. F1-score increases with threshold and reaches its peak when threshold at around 0.1, then decreases with a similar pattern as recall rate. Recall rate is sensitive to low threshold (0 to 0.1), while the precision rate is even more sensitive in this range. After passing the peak point, recall rate becomes more dominative than precision rate and leads F1-score decreasing on higher threshold value. We tend to believe that there is a large portion of 'mixed' data at low threshold region that can't be suitably fitted into our model. So if we adjust threshold, a trade-off between precision and recall has to be made. In our project we are perusing a higher recall rate so we chose a threshold value of 0.1.

RF	Precision	Recall	F1-score	F3-score
Unbalanced - 0 - Tuned	0.96	0.69	0.80	0.71
Unbalanced - 1 - Tuned	0.18	0.70	0.29	0.54
GBDT	Precision	Recall	F1-score	F3-score
Unbalanced - 0 - Tuned	0.95	0.77	0.85	0.77
Unbalanced - 1 - Tuned	0.20	0.60	0.30	0.50
XGBoost	Precision	Recall	F1-score	F3-score
Unbalanced - 0 - Tuned	0.95	0.75	0.84	0.77
Unbalanced - 1 - Tuned	0.20	0.62	0.30	0.51

Fig. 10: Recall and F3 score verses threshold=0.1

In the performance metrics on Figure 10, we included precision, recall, F1 score and F3 score. These results are generated with the same set of parameters, however, we altered the threshold as described above. With a smaller threshold, the precision of predicting class-1 decreased while the recall rate increased. For all three models, we sacrificed precision to get a higher recall rate. In this condition, the F1 score is no longer suitable and representative of the model performance. The F3 score is shown in figure 10 in comparison with F1, which indicate that the RF algorithm has the best overall performance on predicting class 1 on unbalanced dataset. This is due to the fact that RF gives higher recall rate at a small threshold. For the class-0 prediction, both GBDT and XGboost can achieve a relatively good performance.

VII. Conclusion and Future Work

In this project, we investigated a real life classification problem to help the S.F. company identify complaint customer. We also identify the features that are strongly related with complaint to help S.F. company improve the customer experience. The dataset is high dimensional and has class-unbalanced issues. Tree-structured models (RF, GBDT and XGBoost) were utilized for this problem. After parameter tuning, we achieved an AUC score of 0.77 on a balanced set by using GBDT. For the unbalanced set, we further investigated the use of a smaller threshold to achieve a better recall. At our selected threshold of 0.1, RF has the best overall performance.

In the next step of this project, we could further optimize the value of threshold to meet the company's desired precision and recall rate.

VIII. Contribution

Gege Wen: Baseline Model, Feature Selection, Analysis

Yiyuan Zhang: Dataset Pre-process, Feature Selection, Analysis

Kezhen Zhao: Dataset Pre-process, Feature Selection, Tree-structured Models, Anlysis

References

[1] Jernite Y, Halpern Y, Horng S, et al. Predicting chief complaints at triage time in the emergency department[C]//NIPS Workshop on Machine Learning for Clinical Data Analysis and Healthcare. 2013.

[2] Liaw A, Wiener M. Classification and regression by Random Forest[J]. R news, 2002, 2(3): 18-22.

[3] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016: 785-794.

[4] Friedman J H. Stochastic gradient boosting[J]. Computational Statistics & Data Analysis, 2002, 38(4): 367-378.

[5] Tibshirani R. Regression shrinkage and selection via the lasso[J]. Journal of the Royal Statistical Society. Series B (Methodological), 1996: 267-288.

[6] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016: 785-794.

[7] Zhang X, Lu X, Shi Q, et al. Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data[J]. BMC bioinformatics, 2006, 7(1): 197.