Ziwan Peng
pengz21@wfu.edu
11/7/2021

Executive Summary

Problem

As a data scientist in an online job posting service company, I am required to develop a model to identify the fake job posting because fake job postings may lead to some form of identity theft or user reduction due to low credibility of the job postings in our website. That is to say, more fake job postings may cause lower user activity even churns so that the company will meet financial problems like revenue decrease. Along with model development, I also need to construct a profile of what factors will indicate the fraudulent job postings to improve our service. Also, because the data set includes lots of text information such as job description and company profile, we should be careful to deal with text analysis when it comes to feature engineering step.

In this case, it's special that the presence of information matters instead of what content matters. For example, whether there is a department information listed in the job posting matters much more than which department listed in the job posting when we analyze if it's a fraudulent job posting. This rule applies to location. We don't need to specifically consider which location has a higher fraud rate but we should consider whether this job puts the location and how much details it includes. Intuitively, we would think a job posting with a city, state, country is more reliable than a job posting that only tells the country. Therefore, when I move to the data preprocessing step, I should transform those variables in the above way to serve our goal.

Key Findings

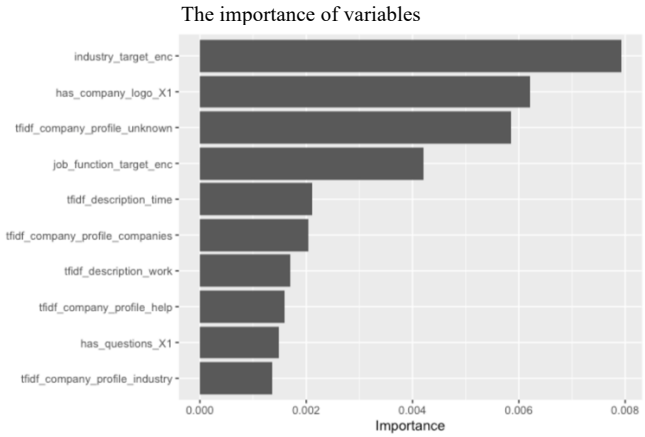
- Presence of a logo is one of the most important predictors to the fake job detection. The probability of fraud for job posting without a company logo is 15.8%, much higher than 2%, the probability of fraud for job posting with a company logo.
- Presence of salary range matters to the fraudulent rate in job posting. It seems counter intuitive that job posting with a salary range are 2 times as likely to be fake than the job posting without a salary range. Because we would think job posting with a salary range looks more 'convincing'. But that maybe the tricky part that real job posting doesn't want to set a specific range less or higher than the candidates' expectations.
- If they have questions is a significant indicator of a legit job posting. The fraudulent rate in job posting without a question is 2 times higher than the fraudulent rate in job posting with a question.

Model Performance Summary & Interpretation

Because our data set is heavily imbalanced, with 5% fraud and 95% legit, we consider the AUC score instead of accuracy as key metrics to measure model performance. The final random forest model has a great performance with extremely high AUC score 97% in test data set comparing with XGBoost model. It means our model can achieve an extremely high classification accuracy in predicting fraud and legit job postings.

Model	Train Accuracy	Test Accuracy	Train AUC	Test AUC
XGBoost	96.7639%	96.7652%	93.2024%	92.6090%
Random Forest	98.8014%	96.9249%	99.9911%	97.1144%

At the feature engineering step, since industry has 130 unique values and job function has 38 unique values, I applied target encodings to deal with high cardinality category issue. In the final model, both the industry and job function are the top 4 important predictors. Because there's a great amount of text information on job description, company profile and benefits, I took a token list with maximum 30 terms and transform them by frequency-inverse encodings to avoid overweight in some terms with high frequency. It turns out key words in job description and company profile play a vital role in prediction on fraud job postings.



I tuned two parameters, number of trees to grow and minimum number of observations to consider for split with k-fold cross validations method. In the end, the combination of 158 as total trees and 18 as minimum observations in a split generated the highest mean AUC score. Therefore, I applied the best performance parameters into the model setting as the final random forest model.

trees	min_n	.metric	.estimator	mean	n	std_err	.config
158	**18**	**roc_auc**	**binary**	**0.9517922**	**5**	**0.00781277**	**Model3**
188	36	roc_auc	binary	0.9508197	5	0.00658578	Model2
170	19	roc_auc	binary	0.9506263	5	0.00747005	Model5
172	39	roc_auc	binary	0.9502026	5	0.00636278	Model4
101	15	roc_auc	binary	0.9439171	5	0.00958961	Model1

Recommendations

- The company can set strict constrains for job postings without a company logo. For example, to put all job postings without a company logo in the end of the job list page so that the job posting has lower click-through rate and much lower chance to be seen by users. It will encourage recruiters to upload their company logo if they want to accept enough applications.
- The company should require more information on job posting when its job function is military and ranching because fraudulent rate of them reach to 100%. For instance, full detailed profile is mandatory. It must illustrate the location, profile, salary range, logo, questions etc. Any missing information is not allowed. The company will refuse to post it on the website if it can't fill out all information.

Detailed Analysis & Steps

Tuning Performance

XGB Model

I tuned two parameters with k-fold cross validation, one is tree depth to decide the maximum complexity of each decision tree and another one is learn rate to determine how much we learn from the loss in the previous prediction. It turns out the 5 as tree depth and 0.021 as learn rate has the highest mean AUC score, implemented in the final XGB model.

tree_depth	learn_rate	.estimator	n	.config	Mean accuracy	Mean roc_auc	std_err accuracy	std_err roc_auc
1	0	binary	5	Model01	0.951	0.5	0.002	0
11	0	binary	5	Model02	0.96	0.883	0.003	0.012
10	0	binary	5	Model03	0.955	0.772	0.003	0.015
10	0	binary	5	Model04	0.96	0.883	0.003	0.011
5	**0.021**	**binary**	**5**	**Model05**	**0.962**	**0.915**	**0.003**	**0.009**
12	0	binary	5	Model06	0.96	0.883	0.003	0.011
9	0	binary	5	Model07	0.96	0.883	0.003	0.011
4	0	binary	5	Model08	0.958	0.869	0.003	0.012
1	0	binary	5	Model09	0.951	0.706	0.002	0.024
8	0	binary	5	Model10	0.951	0.5	0.002	0

Random Forest Model

I tuned two parameters with k-fold cross validation, one is trees to decide the number of trees to grom and another one is minimum number of observations to consider for a split. It turns out the 158 as total trees and 18 as minimum observations in a split has the highest mean AUC score, implemented in the final random forest model.

trees	min_n	.metric	.estimator	mean	n	std_err	.config
158	**18**	**roc_auc**	**binary**	**0.9517922**	**5**	**0.00781277**	**Model3**
188	36	roc_auc	binary	0.9508197	5	0.00658578	Model2
170	19	roc_auc	binary	0.9506263	5	0.00747005	Model5
172	39	roc_auc	binary	0.9502026	5	0.00636278	Model4
101	15	roc_auc	binary	0.9439171	5	0.00958961	Model1

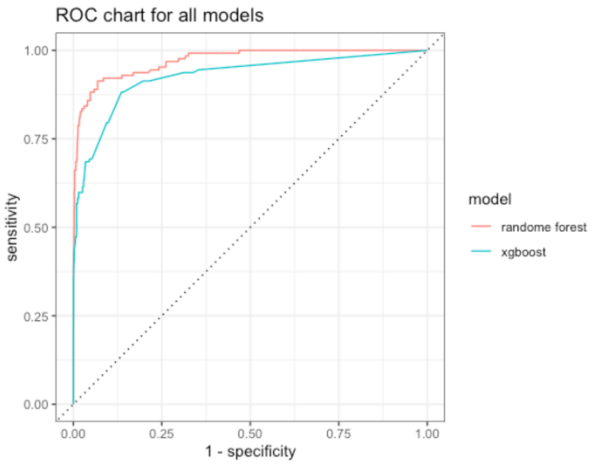
Metrics

After I put the best parameters in each model and fit with train data and test data, it appears that random forest has a better performance both in train data set and test data set.

Model	Train Accuracy	Test Accuracy	Train AUC	Test AUC
XGBoost	96.7639%	96.7652%	93.2024%	92.6090%
Random Forest	98.8014%	96.9249%	99.9911%	97.1144%

ROC Curve

From the ROC chart, we can observe that with a same sensitivity, the random forest has a lower false positive rate(1-specificity). With a same false positive rate, the random forest has a higher sensitivity. Therefore, the random forest model has a better performance than xgboost model in both ways.


ROC chart for all models
sensitivity
1 - specificity
0.00
0.25
0.50
0.75
1.00
model
randome forest
xgboost