Glassdoor ML Xuelong Wang

Xuelong Wang

2/20/2019

Goal and background

The Goal is to predict the apply rate for a given job description and applicants features.

Data

The apply data set has 1200890 observations. There are 7 features which are used to predict the apply rate. Most of them are measuring how close the job descriptions with the applicants' features.

Data clean and preprocess

Before doing any real analyses, I obtain some descriptive statistics of the data and do some adjustment to remove the inconsistency among the data. There are certain amounts of NA's in the `title_proximity_tfidf`, `description_proximity_tfidf` and `city_match`, which all are removed. The summary statistics also show that those features have a different range from 10 to 100. I apply a Max-min transformation to rescale them to 0-1 range so that we could interpret the coefficients more easily.

Unbalanced class

Based on the apply data, the chance of a person click the apply button is 0.09, which is small. This is kind of make sense because we will probably search for more positions than we actually apply for. However, it's not easy to predict the apply rate well under this unbalanced situation. The dominated class (in our case is not apply `apply = 0`) will affect the models so that most of them will tend to predict an observation as "not apply". In order to handle that, I use the up-sampling method to resample the minority class so that both classes will have the same number of observation.

class_id

The `class_id` is the job description id. There are duplicated values in the `class id` columns, which means a job has been visited more than one time, likely, by different people. If a job is frequently visited than it's probably popular. Therefore, it's natural to assume that there are more people to apply for that job. So I replaced the `class_id` with the visiting frequency of that job. and then use the frequency as an 8th feature to fit the models. It turns out that improves AUC a little.

Models

I have tried two commonly used models which is Logistic regression and Support Vector Machine (SVM). Both of the methods can be used to solve the prediction problem in the binary case. The evaluation method is AUC which is more informative than others in the unbalanced case.

Result and conclusion

Training result

	SVM	Logistic
AUC	0.54	0.55

The result is based on the average of 3-fold cross-validation. The results of those two models are closed. Logistics's AUC is little higher than SVM and also for the interpretation, we choose the logistic regression model.

Testing result

	Logistic
AUC	0.542

The testing result is lower than the testing result but not much.

Coefficients

title_	description	main_query	query_jl	query_title	city_match	job_age	freq
0.41	-0.62	0.31	0.9	1.9	0.04	-4.55	0.16

Based on the estimated coefficients, the `job_age_day` seems to have a large negative influence on the log ratio of the apply rate. This could be interpreted as: the longer a job is posted on the list the less it will be applied.

It seems that using the class id's frequency does not improve the prediction result a lot. One possible reason is that the freq is related to also related to job_age_days because the longer it posed the more it is visited.

Future work

If I have more time, I would like to explore several things as follows:

1. feature selection and transformation: I would like to get a more details description of the features so that I could get a deeper understanding of them and find an efficient way to use the features.
2. model selection: fit different models with grid search parameters
3. unbalanced method: I would try a different approach