Mithil Vijay Gotarne
mithilvijay.gotarne@utdallas.edu

Prediction of Apply Rate

1. Introduction

The problem of interest is the prediction of apply rate. Imagine a user visiting Glassdoor, and performing a job search. From the set of displayed results, user clicks on certain ones that she is interested in, and after checking job descriptions, she further clicks on apply button therein to land in to an application page. The apply rate is defined as the fraction of applies (after visiting job description pages), and the goal is to predict this metric using the dataset provided.

2. Problem Definition and Algorithm

2.1 Task Definition

We need to find out apply rate of the jobs posted on Glassdoor. For performing this task we have a dataset with 1,200,890 rows where each row in the dataset correspond to a user's view of a job listing. It has 10 columns as described below.

1) title_proximity_tf_idf: Measures the closeness of query and job title.
2) description_proximity tf idf: Measures the closeness of query and job description.
3) main_query_tf_idf: A score related to user query closeness to job title and job description.
4) query_jl_score: Measures the popularity of query and job listing pair.
5) query_title_score: Measures the popularity of query and job title pair.
6) city_match: Indicates if the job listing matches to user (or, user-specified) location.
7) job_age_days: Indicates the age of job listing posted.
8) apply: Indicates if the user has applied for this job listing.
9) search_date_pacific: Date of the activity.
10) class_id: Class ID of the job title clicked.

We will use the first 7 columns. Use these as features to predict the 8th column, "apply". The will try to incorporate 10th column and check if it's involvement in training can get is us better results.

2.2 Algorithm Definition

This is basic classification problem where we have to predict if the person applies or not. For this purpose we will use a binary classifier to predict the apply feature. For this purpose will use Logistic Regression, Gradient Boosting and Support Vector Machines. The purpose of using a logistic regression and SVM is the it is a basic and best binary classifiers. The extensive data exploration has shed some light on the fact that data available has around 252,571 missing values and imbalance of training data (i.e. 90% class 0, 10% class 1 of apply column). The intent of using XGBoost is that it handles imbalanced and missing data very well.

3. Experimental Evaluation

3.1 Methodology

3.1.1 Exploratory Data Analysis

Some EDA is done to find out missing values, get histograms of all the columns and correlation matrix was plotted to determine if there were any related features. Some of the features are continuous values while some are categorical discrete values accurate correlation wasn't calculated. Because of EDA is came to notice that data was divided into 90% (class 0) - 10%(class 1) ratio and had around 252,571 missing values.

3.1.2 Preprocessing

a. **Normalization of Data**
   Non-categorical columns were normalized with mean 0 and std deviation 1.

b. **Data Splitting**
   Training dataset is the data with dates 01/21/2018-01/26/2018, and testing dataset of date 01/27/2018.

c. **Oversampling (SMOTE)**
   Because of unbalance in the dataset, Synthetic Minority Over-sampling Technique (SMOTE) was used to make dataset of 60:40 ratio.

3.1.3 LogisticRegression

Logistic Regression with balanced weights parameter was used as a model

3.1.4 XGBClassifier

XGBClassifier with positive weight scaling was used as a model

3.1.5 SVC

Basic SVC ran for long time without halting due to computational resources limit.

3.2 Results

AUC Score was the right metric to measure performance. We used the model to predict probability of class 1 to create AUC Score. Apply rate is the ratio of applies after visiting, probability of class 1 exactly predicts that. This probability of class 1 every test sample is used to construct ROC - AUC Score.

ROC - AUC SCORE

Classifier	**without class_id**		**with class_id**	
	without SMOTE	**with SMOTE**	**without SMOTE**	**with SMOTE**
LogisticRegression	0.5884	0.5887	0.5880	0.5835
XGBClassifier	0.6068	0.5579	0.6224	0.5876

We can see from the above results that with class_id AUC score increased. class_id had 157 unique values, which means it helped it data segmentation and can be used as categorical value.

The accuracy metric was not used in this case, as the imbalanced data will almost make model predict majority class, which will always give accuracy around 90%. This accuracy of 90% is because of majority class and minority class if always misclassified. The accuracy metric here gives us false performance of the model. Precision, recall and f1-score are better metrics to use in such case.

4. Future Work

Due to computational and time limitations grid search of hyperparameters and for exploring other imbalanced-learn algorithms was not possible. Future work can be to explore these options. More domain knowledge on class_id could have helped use to segment data more efficiently as 157 are many categories to work on.

5. Conclusion

Thus we can say that XGBoost worked well with predicting apply rate and inclusion of class_id definitely improved our model.