Report On the Prediction of Apply Rate

Ramesh Paudel
rameshpaudel_4@yahoo.com
1314 W Broad St. Apt G1
Cookeville, TN 38501

For predicting the job apply rate for user visiting Glassdoor and performing a job search, following steps were followed.

I. DATA PREPROCESSING

All the experiments were done in python using the sci-kit learning tool's. The data was split into test and training set based on "*search_date_pacific*" column where training set consist of data rows between 01/21/2018-01/26/2018, and test set consist of rows from 01/27/2018. The distribution of each feature was shifted to have a mean of zero and a standard deviation of one (unit variance), because distance based classifier like SVM and KNN calculates the distance between two points, and if there is a feature with broad range of values, the distance will be governed by this particular feature. Based on our observation we found some features like *title_proximity_tfidf* and *job_age* has broad range of values. Therefore, data scaling was done so that each feature contributes approximately proportionately to the final distance. Algorithm were run on the scaled data using 10-fold cross validation.

II. METHODOLOGY

In order to represent a cross-section of the most common machine learning techniques we will use following approaches for experimentation.

- Logistic Regression
- Artificial Neural Network
- Naïve Bayes
- Decision Tree
- Random Forest
- K-Nearest Neighbor (KNN)
- Bagging (Decision Tree)
- Ada Boosting

This will provide us with a comprehensive analysis of traditional machine learning techniques for predicting apply rate in Glassdoor job search. Logistic regression was applied on the dataset with first 7 columns and the result showed auc of 50%. Breaking down the recall for both *apply* and *no-apply* class, it showed that *apply* had a recall of 0% (shown in Table 2) meaning the model built was not able to classify *apply* class. This was because training set had imbalance data (*apply* to *no-apply* ratio ≈ 1:10) meaning the prediction decision of model was dominated by *no-apply* class. To deal with imbalance data, three techniques down-sampling *no-apply* class, up-sampling *apply* class, and weight adjustment were tested. Weight adjustment had the better performance so for algorithms that provide weight adjustment (Logistic Regression, Decision Tree, Random Forest, and Bagging) this technique is used while all other algorithms were run on original data.

Algorithm	Precision	Recall	F1-Score	AUC
Logistic Regression	.86	.60	.69	.57
Decision Tree	.84	.83	.83	.50
Random Forest	.84	.91	.87	.50
KNN	.84	.91	.87	.50
Bagging	.84	.48	.58	.50
Ada Boosting	.83	.91	.87	.50
Artificial NN	.83	.91	.87	.50
Naïve Bayes	.85	.90	.87	.52

Table 1: Analysis I result

	Precision	Recall	F1-Score
no-apply	.91	1.0	.95
apply	0.0	0.0	0.0
total	.83	.91	.87

Table 2: Logistic Regression Performance on original data

III. ANALYSIS I

The result of all algorithms using first 7 features is shown in Table 1 while AUC curve is shown in Figure 1. Considering the results, out of the chosen eight algorithms, Logistic Regression has the best performance in terms of AUC (57%), Naïve Bayes comes 2nd with AUC (52%), while all other have AUC 50%. Though, F1-score and recall for logistic regression is low, breaking down these metrics for both class, logistic regression had superior performance for both *apply* and *no-apply*, while other algorithm could only classify majority class *no-apply*. This indicates *no-apply* class is easier to recognize, possibly as a result of having more data points (*apply* to *no-apply* ratio ≈ 1:10).

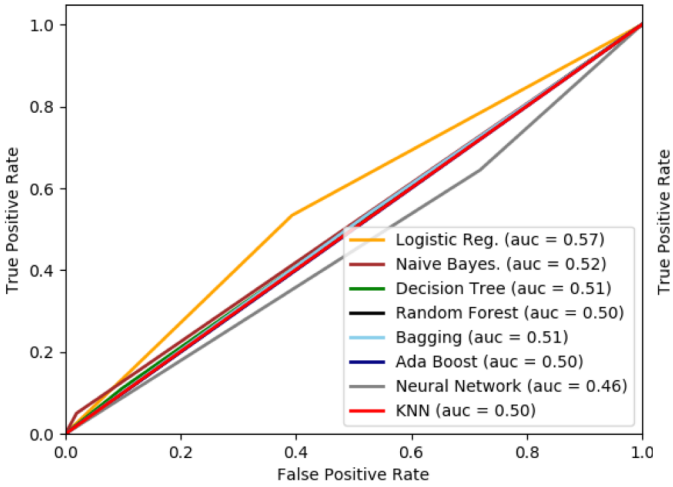

Figure 1: Analysis I AUC

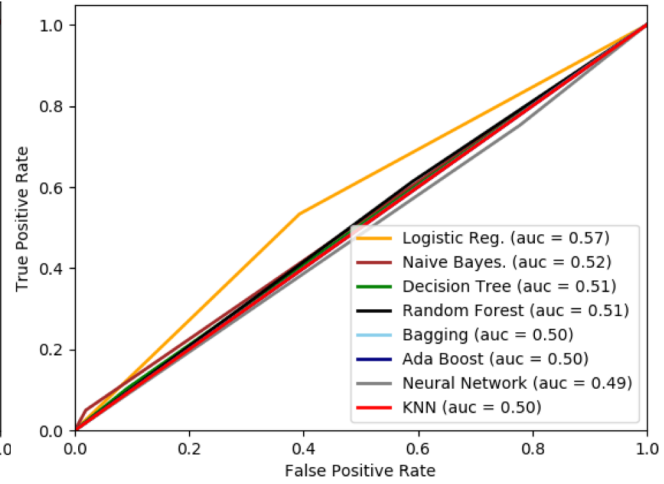

Figure 2: Analysis II AUC

III. ANALYSIS II

The experimentation was further extended by adding 2 more features *u_id* and *mgoc_id* as input. The AUC curve for all algorithm using 9 features is shown in Figure 2. Considering the results, out of the chosen eight algorithms, Neural Network and Random Forest had the slight improvement in their performance in term of AUC, while all other algorithm didn't show any improvement.

IV. CONCLUSION & FUTURE WORKS

Since analysis I has 7 features all of which have continuous values. Naïve Bayes, Neural Network and Logistic Regression are supposed to have better performance on these kind of data which was reflected on the results. For Analysis II, adding 2 other feature helped improving the performance of Neural Network, while it didn't have any improvement in other algorithms.

All the experiments were performed using default parameter for all algorithms. Some form of parameter tuning could have increased the performance which we would like to do in future. I would like to test other technique to deal with imbalance class in future. I tried support vector machines, but it took long time to finish and I was not able to include the results, so in future I would definitely try analyzing the performance of support vector machine.