ON THE PREDICTION OF APPLY RATE

INTRODUCTION

The dataset consists of a 10-dimensional feature set, having numerical and categorical values. The task is to predict the Apply Rate, given the features that are mostly scores of interest like closeness of query and job title, closeness of query and job description etc. From intuition, each of the features will affect the prediction of the Apply Rate. From an applicant's perspective, the most important features contributing to apply rate would be the closeness of query to job title and job description. Also, a direct correlation should exist between the popularity of the query and job title pair or the popularity of the query and job listing pair, with Apply Rate.

INITIAL EXPLORATORY ANALYSIS

1. **Removing Duplicates and Conflicting Data:** Removed Duplicate data points from the data. Data was reduced by around 8%. Further, I found that there was conflicting data, i.e. rows having exactly same set of features from 1^{st}-7^{th} column, but labeled differently. I removed the duplicate 0 labels because Class 1 is imbalanced and removing Class 0 points wouldn't skew the data much. This reduced further 8% of the data. Total ~15.5% of data was reduced after removing duplicates.
2. **Recognizing the Important Features and Removing the Redundant Features:**
   - Firstly, on evaluating the city match feature column, I found about 30% of the values as NaN. Since it's a binary feature and the distribution of missing values is about the same for both the classes, I chose to drop it from further evaluation. (Plotted a Box-plot to verify importance in classification)
   - On plotting the correlation values and scatter-plots between the features, I found that the features having a substantial value of correlation with 'apply' were: main_query_tfidf – 0.038, query_jl_score – 0.068, query_title_score – 0.074, title_proximity_tfidf - 0.058.
   - Further, job_age_days is negatively correlated to apply, which is intuitive since as the job age increases, usually there are lesser and lesser applications on the job.
3. **Handling Missing Values:** Since removing rows with missing values can be too limiting on predictive modeling problems, an alternative is to impute missing values in the entire dataset. I chose to Impute the missing value by the median value of the column because the idea is that the value in the middle of the range will not significantly affect prediction.
4. **Class Imbalance:** The ratio of the two classes is about 1:9 in the training data and same is reflected in the validation set when calculating cross-validation scores. To work around this issue, I tried to implement resampling techniques, namely SMOTE (Oversampling) and Random Undersampling. Random Undersampling performed better than SMOTE on the test data.
5. **Scaling:** Since the score measures are scaled differently, the feature having higher value may bias the model prediction. Hence I implemented StandardScalar Function so that each feature has a distribution with 0 mean and unit variance.

FEATURE ENGINEERING

As an initial feature engineering decision, I assume we would get better insights from job_days_age on converting it into categorical levels such as Today(Same day), within_a_day(Next day), within_two_days, within_a_week, within_a_month, within_a_year and more_than_a_year. Logically, better correlation to binary labels is observed at categorical level than for numerical features. For example, it is seen that people practically apply the most on the next day and within the week and month as these columns have higher correlation than others. This can be analyzed as, the same day information about the job posting is not spread properly. The next day, the mails go out hence the apply rate is maximum the next day. Further, some people can save data and work on the application and hence apply within the week or next week. The number of applications for a post more than a month old is practically very less and within_a_month and within_a_year columns have substantial negative correlation to the Apply Rate.

Also, binarizing features can introduce nonlinearity to linear models which might help increase prediction accuracy.

MODEL SELECTION

Implementing basic algorithms at first and testing the performance, I didn't get substantial AUC score above 0.5. Since decision trees are robust to scale of the features in the dataset and missing values, I decided to implement Random Forest Classifier and it gave substantially good performance on the test data. Hence, I decided to implement Gradient boosting Classifier also as a superior method to RFC and it gave better performance on test data.

Since Intuitively this is a non-linearly separable data, we cannot use LinearSVC. But running the data through LinearSVC with class_weight = balanced (to handle class_imbalance), without Class_ID gave AUC of 0.53 while with Class_ID gave AUC of 0.58.

The AUC metric of the classifiers tried can be seen below:

Model	AUC without Class_ID with imbalanced data	AUC without Class_ID with SMOTE	AUC with Random Undersampling
Gradient Boosting Classifier	0.5008	0.5775	0.5825
Logistic Regression	0.5000	0.5695	0.5709
Random Forest Classifier	0.5084	0.5212	0.5415

Also, Incorporating the New features in place of Job_age_days, increased the AUC for gradient boost classifier from 0.5719 to 0.5744. Hence, the feature engineering helped classification.

INCORPORATING CLASS_ID

Yes, it is possible to segment the data using the Class_ID as there are 157 Class_IDs among which the Apply and Non-apply labels are distributed. It gives an insight on the popularity of the job listing. Initial Correlation plot of Class_ID indicated high correlation with the query_title_score and query_jl_score intuitively meaning that the more the popularity score, more is the number or count of Apply and Non-apply labels for the Class_Id i.e more people open the job description page.

To have a good data for prediction, I converted the Class_ID into an important feature column, the Apply_Ratio of each Class_ID signifying each Job title. From correlation plot, the Apply Ratio gives the highest correlation of 0.098 with Apply Data. And as can be seen from the results, incorporating the Class_ID improved the AUC scores.

Model	AUC with Class_ID with imbalanced data	AUC with Class_ID with SMOTE	AUC with Class_ID with Random Undersampling
Gradient Boosting Classifier	0.5004	0.5849	**0.5978 ~ 0.6 AUC**
Logistic Regression	0.5000	0.5883	0.5886
Random Forest Classifier	0.5118	0.5325	0.5533

CONCLUSION

From the above analysis, I can conclude that the best performing classifier on the data is Gradient Boosting Classifier using Random undersampling: 0.6 AUC. Class-id is an important feature contributing to the Apply Rate. And, the most important features are: main_query_tfidf, query_jl_score, query_title_score & title_proximity_tfidf.

FUTURE IDEAS: IF MORE TIME WAS AVAILABLE

- Implement cost-sensitive learning algorithm and use kernel method for non-linear transformation of data. For example classification using SVC with RBF Gaussian kernel for separating data in high dimensional space. SVC with non-linear parameterization i.e. RBF kernel was computationally expensive and failed to output any results within the stipulated time.
- I would implement PCA to convert the data into 2D to visualize any class separability and reduce the number of affecting features.
- Implement Shallow and Deep Learning models.
- More Feature Engineering. Can probably add query_title_cty_score and query_description_city_score which might give us more correlation with apply rate.
- Implement Outlier detection algorithm to further clean the data before model fitting.
- Since Gradient Boosting Classifier gave the best accuracy, Implement Gradient Boosting with Tuned Hyperparameters using the undersampled dataset with features related to class-ID.
- Intended to plot Violin plots that guide us in feature engineering and selection by revealing the variables that best separate the two classification outcomes that best separate the two classification outcomes.