CS 565 Data Mining, Fall 2017 Instructor: Prof. Terzi Evimaria

Project 2: Kaggle Competition

Dec 10th, 2017 Tian Zhang

(My Kaggle username is **ztian0911**)

1. Dataset:

In this project, we are supposed to implement prediction models to a user review dataset of flight. The training and testing dataset contain the same attributions, including flight information, user type, user ratings regarding to eight aspects of flight service. Meanwhile we need to use these attributes to prediction whether a user will recommend the flight or not.

2. Data Prepossessing:

(1) **Matrix Completion**: In the original dataset, most of the attributes do not have complete record. Thus, we need to use some strategy to fill these missing values. In following paragraphs, I will illustrate my strategies for each attributes including categorical and continuous.

a. **cabin_flown/ type_traveller**: For these two attributes, I try to complete the missing value by using a new category "Unknown" to present. And then I convert them to dummy variables.

b. **Eight Rating Attributes**: For these attributes, I try three different methods to fill the missing values: 0, median and mean. Comparing the prediction result, I found that using the mean of each column to present to missing values results highest accuracy.

c. **date**: For data, I convert them to numerical month data (1-12) firstly, and then convert the month to four dummy variables(corresponding to Winter, Spring, Summer and Autumn). I believe the season might be an important feature to influence the recommendation of user, as it is related to weather directly.

(2) **Review Content Analysis**: In my opinion, the sentiment of user reviews will influence the recommendation directly. At first, I try to use Word2Vec strategy to embedding the reviews to vectors and then fit them to RNN with LSTM structure, but it takes me plentiful time to train and tune. Thus, finally, I used a simple way to present user reviews to binary vectors. I try to count the most frequent 300 words and then choose 20 sentiment-presented words as dummy variables to match the existence of corresponding words in each review.

3. Training:

In general, I tried three algorithm for prediction, including linear regression, decision tree and random forest. For linear regression, I tried to figure out the contribution of each attributions, which can be treated as a feature engineering. After choosing the appropriate attributes for building prediction model, I implemented decision tree to find out the relationship between variables in a top-down tree structure. But I found the following two problems with decision tree that might leading to bad performance in our dataset.

- It is hard to find a global optimal. Specifically, at each step the algorithm chooses the best result. However, choosing the best result at a given step does not ensure you will be headed down the route

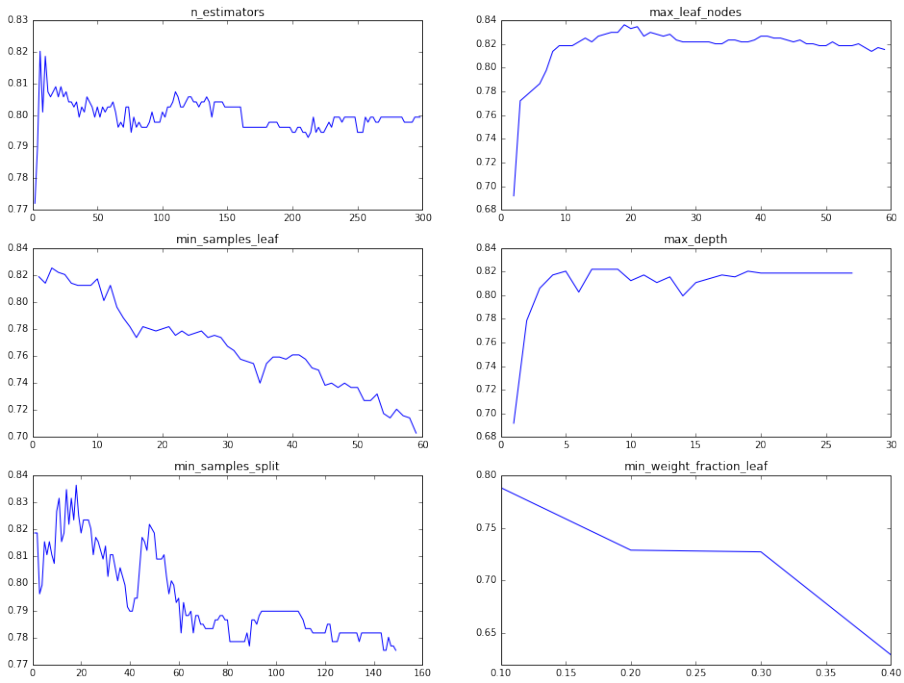

Figure 1: Tuning Parameters for Random Forest

that will lead to the optimal decision when you make it to the final node of the tree, called the leaf node.

- I find that using decision tree will lead to over-fitting more easier, which means the accuracy of training set is high but low for testing set.

According to the above problems for decision tree, I used random forest instead. As we known, random forest can be regarded as a improvement of decision tree, because a random forest is simply a collection of decision trees whose results are aggregated into one final result.

For implementing the random forest, I utilize the **sklearn.ensemble.RandomForestClassifier** package. **sklearn** is a really convenient package for machine learning. It offers us some parameters for tuning in the training part. Thus, I believe, after building feature matrix, fine-tuning the classifier is the most important task for building next-step prediction model.

In the tuning step, I set each parameter in a pre-defined range, and then use a for-loop to explore the most efficient parameter combination. For evaluation the performance of each parameter combination, I use accuracy curve as above graphs(Figure 1).

And I find some tricks in the process that **"n_estimators"** is not really worth optimizing. The more estimators you give it, the better it will do. 500 or 1000 is usually sufficient.I choose 1000. **"max_features"** is worth exploring for many different values. It may have a large impact on the behavior of the RF because it decides how many features each tree in the RF considers at each split. **"criterion"** may have a small impact, but usually the default is fine. .

Finally, for the testing dataset, my random forest performed 0.95801 accuracy.(For details, Please viewing the markdown in Jupyter Notebook file.)