Class Project Report

1. Data Preparation

In this section, I will discuss about how I prepare data for this project. First, I import train_label.csv and then split "user_id#merchant_id" into "user_id" and "merchant_id". Meanwhile, from the train_label.csv I store "proba" into a numpy array "train_prob". Then I import user_log.csv and user_info.csv. For user_info.csv, I will import age_range and gender then based on the file to match user_id and these two to form the feature "age_range" and "gender". For user_log.csv, I will match user_id, merchant_id and the features in this file, where I will just import the transactions whose action_type is not zero. Then I find all time_stamp here is "1111", so I will remove the feature time_stamp.

For the future prediction, I have to create some new features, here I first create "buy_before" and "times" features. "buy_before" is a binary feature, where 1 represents users who make some nonzero action_type for the same item and 0 for not. "times" represents users' on the same item action_type weight before November 11, which 0 for none action_type before, 0.5 for action_type 1, 1 for action_type 2 and 1.5 for action_type 3, then for action_type nonzero ones, adding it up to form this feature. After that I also create "same_cat" and "times_cat" features, which means I also focus on users' action on the same category before. So these two creation are similar to "buy_before" and "times" and the only difference here is I focus on the category. Finally, I want to find the relationship between merchants and items, so I create "merchant_s_item" feature to show whether this merchant sells the same item before but all is 1, so this feature creation fails.

Thus, here I list all labels I will use in an order(number represents column):

1: user_id 2:merchant_id 3:item_id 4:cat_id 5:brand_id 6:action_type 7:age_range 8:gender 9:buy_before 10:times 11:same_cat 12:times_cat 13:merchant_s_item 14:proba.

Column 3 to 13 are features for prediction and Column 14 is the result for training data, thus for testing data this column is null. So based on this method, I create two new csv files, newtrain.csv file for storing train data and newtest.csv for storing test data.

2. Models

The instructor suggests us to use ensembling method to build models, thus here I use two emsembling models for working and comparing. But first all I have to do is using feature selection method to pick some features for predicting. Here I import sklearn package and use univariate feature selection method and f_classif value for this process. I set k to 3, 4 or 5 for next process.

Then I use randomforest emsembling method to build the prediction model. I firstly sample 1/3 train data for testing and rest data for building the model. And I find the accuracy among these three models are all relatively high, which all between 0.93 and 0.94, and every time 4 features' model is always highest. So I use all train data for test data and write the result into the submission.csv file and submit to the Kaggle, where I find 3 and 4 features' model are better than 5 features' model, whose log loss is nearly 0.7.

Finally I also use Adaboost method to build the prediction model and select 4 features, then do the same process again and find although the accuracy for train data is good, for test data it is worse than randonforest one.

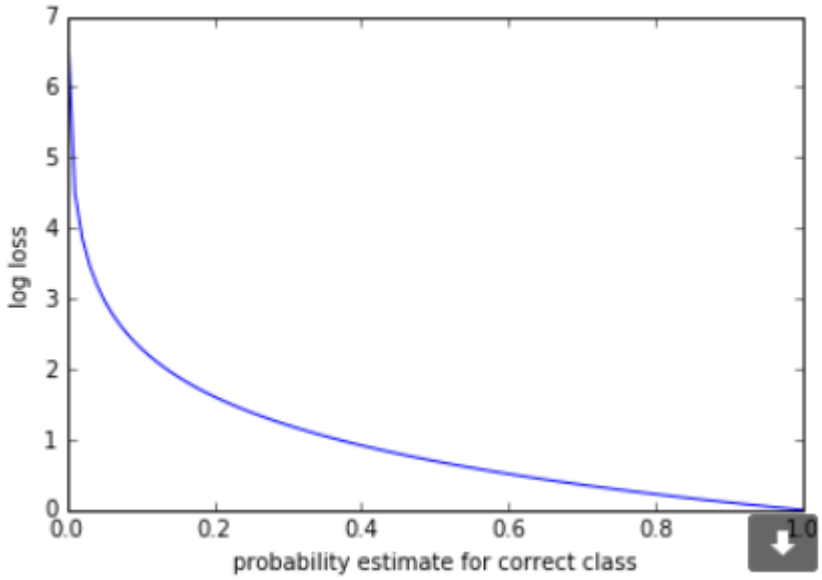
I think the reason why randomforest model is better than Adaboost one is that Adaboost model is sensitive to noise and outliers, while randomforest model is strong enough to overcome this disadvantage. And when I select 5 features in the model, although it fits train data well, it not fits test data because of some disturbing features causing the overfitting problem. And when I choose too few features, like just 1 feature, it will cause the underfitting problem. So I think select 3 or 4 features is suitable.

3. results

Here I choose 4 features' randomforest result, which selects buy_before, times, same_cat and times_cat features, for my final result and also I will insert some screenshots. For train data, I replicate five times and its accuracy scores(randomforest score method to show mean accuracy) are as follows:

```
>>> ================================ RESTART ================================
>>>
0.938992958071
>>> ================================ RESTART ================================
>>>
0.941501357758
>>> ================================ RESTART ================================
>>>
0.939614304782
>>> ================================ RESTART ================================
>>>
0.937957380218
>>> ================================ RESTART ================================
>>>
0.939729368988
>>>
```

For the final test result, its log loss is 0.24214.



I think this result is reasonable one for prediction, because I think it will classify test data into right classification in most cases. But I think there still leaves some improvements. The most important one here is I fail to build the suitable relationship between merchants and items through transactions, which makes my prediction rank not high.

4. Lesson learned

I think from this class project I have learned how to use feature engineering to create features on my own for classification based on several data files, not just wait for the train data which is created by others and good enough to use. Meanwhile, I also learn why ensemble methods are better than some classification algorithms I have acquired from the class, and through comparing two ensemble methods, which one is better. Finally, because I use Python to write the program, I am more familiar with Python language and numpy array. Based on that, I can more easily analyze and write programs for data mining in the future.