CS150A Database
Course Project

Wu Yuheng
ID: 2020533018
wuyh4@shanghaitech.edu.cn

Li Zongze
ID: 2020533177
lizz@shanghaitech.edu.cn

Guideline

Compared with developing a novel machine learning algorihtm, building a machine learning system is less theoretical but more engineering, so it is important to get your hands dirty. To build an entire machine learning system, you have to go through some essential steps. We have listed 5 steps which we hope you to go through. Read the instructions of each section before you fill in. You are free to add more sections.
If you use PySpark to implement the algorithms and want to earn some additional points, you should also report your implementation briefly in the last section.

1 Explore the dataset

Instruction:
Explore the given dataset, report your findings about the dataset. You should not repeat the information provided in the 'Data Format' section of project.pdf. Instead, you can report the data type of each feature, the distribution of different values of some important features(maybe with visualization), is there any missing value, etc

Your work below:

During the exploration of the data, we found that in the 19 coloumns of the data, there are a series of numerical data such as 'Problem View' and a series of categorical features such as 'Anon Student Id'. Specially, for the last two coloumns, which is 'KC' and 'Opportunity', they have a combination of several features and is separated by the wave sign .

2 Data cleaning

Instruction:
Some people treat data cleaning as a part of feature engineering, however we separate them here to

make your work clearer. In this section, you should mainly deal with the missing values and the outliers. You can also do some data normalization here.

Your work below:

First, when dealing with the value NaN, we adopt **math.isnan** to identify it. And at all circumstances, when a NaN may affect the code, we will replace it with the according mean value of that coloumn.

Second, for data normalization, we use **encoder.fit_transform** from **sklearn.preprocessing** which is used to convert string to numbers. It is worthy noticing that since the projection created by one call of the function is not unique, we should use the raw data when searching for certain conditions.

Third, for the normalization of 'KC' and 'Opportunity', we first split these two values according to the wave lines that sperates them, then we make several copies of that line according to the numbers of features in that 'KC' and 'Opportunity'. We fill in each line with the splitted values.

3 Feature engineering

Instruction:
In this section, you should select a subset of features and transform them into a data matrix which can be feed into the learning model you choose. Report your work with reasons.

Your work below:

First, it is important to mention that we notice the fact that in the train data, several variables are all NaN, including 'Step Start Time', 'First Transaction Time', 'Correct Transaction Time', 'Step End Time', 'Step Duration (sec)', 'Correct Step Duration (sec)', 'Error Step Duration (sec)', 'Incorrects','Hints', 'Corrects'. Considering this, when constructing our features, we should hardly take these parameters into consideration.The parameters left should be 'Anon Student Id', 'Problem Hierarchy', 'Problem Name', 'Problem View', 'Step Name','KC(Default)' and 'Opportunity(Default)'.

Second, we calculate CFAR, which is Correct First Attempt Rate, configured by the formula below:

$$CFAR_{parameter} = \frac{\#parameters|CFA == 1}{\#parameters}$$

Parameters mentioned above includes 'Anon Student Id', 'Problem Hierarchy', 'Problem Name', 'Problem View', 'Step Name','KC(Default)' and 'Opportunity(Default)'.

Third, we calculate the number of KC, mean value of 'Opportunity' and min value of 'Opportunity'. For example, if we have a line in which 'KC' equals to $'A \sim B \sim C'$ and 'Opportunity' equals to $'1 \sim 2 \sim 3'$, then the number of 'KC' would be 3, mean value of 'Opportunity' would be 2 and min value of 'Opportunity' would be 2.

4 Learning algorithm

Instruction:
In this section, you should describe the learning algorithm you choose and state the reasons why you choose it.

Your work below:

The models' performance are that:

RandomForest : 0.433

LogisticRegression : 0.447

Decision Tree : 0.448

AdaBoost : 0.463

So we choose RandomForest as our learning algorithm.

5 Hyperparameter selection and model performance

Instruction:
In this section, you should describe the way you choose the hyperparameters of your model, also compare the performance of the model with your chosen hyperparamters with models with sub-optimal hyperparameters

Your work below:

We use **GridSearchCV** to search for the best parameters for the model. The train set includes 'Problem Name', 'Step Name', 'KC(Default)', 'Opportunity(Default)','rsid','rs_name','rkc','rsid_pname', 'min_opp', 'mean_opp'. Our param_grid includes 'n_estimators', 'max_features', 'max_depth' and 'bootstrap'. The result of the search is presented in the following table.

n_estimators	150
max_features	0.7
max_depth	5
bootstrap	True

6 PySpark implementation (optional)

For pyspark implementation, we write five functions.

First is pre-process function, we manipulate one feature each time, and use select(),distinct(),collect() function to to get the no-repeat element list, then we use drop() function to drop the 'Step Start Time', 'First Transaction Time', 'Correct Transaction Time', 'Step End Time', 'Step Duration (sec)',

’Correct Step Duration (sec)’, ’Error Step Duration (sec)’ these “useless” columns for original train and test set, use union() function to union them by index, finally reset these manipulated columns.

Second and third are the functions to calculate cfa_rate, they are auxiliary functions, return the number of right correct first attempt value(that is equal to 1), they are aim to process train and test set respectively.

Last two functions are main function to calculate cfa_rate, we manipulate one feature each time and first use groupby() function to get “sets of the features” then use count() function to count corresponding numbers, and we use auxiliary functions above to get right numbers in sum, finally we can get corresponding ratio. And last step is join them, we use join() function and use with drop() and select() (.join(col, col[fea] == right[fea]).drop(col[fea]).select(fea,(right[’count’] / col[’count’])) to join them.