RIIID! ANSWER CORRECTNESS PREDICTION

Yang-Sheng Chao*, Hui-Chi Kuo †, Yen-Ru Chen‡, Chun-Han LI§
Student ID: *106062132, †106062212, ‡106034026, §109062595
Group 9 of Kaggle Team
2020 ISA 5810 Data Mining

***Abstract*—In order to let most students have the access to the personalized learning, the participants of this Kaggle Competition need to create algorithms for "Knowledge Tracing", the modeling of student knowledge over time. Based on the behaviors of the students do when learning online, which were provided by the dataset, we need to create a model to predict whether the students will answer the question correctly.**

I. Introduction

Since we have to use the Kaggle API (online Notebooks) to write the submission, we need to meet the limitation of the online Notebooks:

- CPU Notebook <= 9 hours run-time
- GPU Notebook <= 49 hours run-time
- TPU Notebook <= 3 hours run-time

Hence, we need to do the *Data Preprocessing* in advance to load the data successfully. Since we can only see the result of the public leaderboard before the competition was over, which contains only part of the dataset, the robustness of our model would be our main concern to avoid the problem of overfitting. We've tried two ways to do the data preprocessing and model traing to get the best result.

II. Methodology

A. Model 1

1) Data Preprocessing: Since the data is extremely huge, we have to transfer the data type while loading it. Especially in $train.csv$, we transfer $prior\_question\_had\_explanation$ from object to boolean, which can reduce the memory usage from 3594972816 to 101230332 bytes **(97% reduced)**.

2) Data Preparation: Before feeding the data into the model, we have done some feature selections, which not only can help to reduce the dimension but also reduce the training time and memory usage. Eventually, We choose $prior\_question\_had\_explanation$, $prior\_question\_elapsed\_time$, $answered\_correctly$ as our features. In fact, in our process, with $user\_id$ and $content\_id$, we transfer $answered\_correctly$ into four feature which are $user\_questions\_amount$ (The number for a user has answered), $user\_correct\_rate$ (The correct rate for a user), $questions\_answered\_amount$ (The number for a question has been answered), $questions\_correct\_rate$ (The correct rate for a question). Then, as the data is too huge, we only sample **40000000 data (40%)** and fill the missing value by the mean of that feature except for $user\_questions\_amount$ and $questions\_answered\_amount$, for these two features, we fill the missing value by 0 instead of the mean since we actually don't know how many time they have answered or they have been answered(In fact, the testing data contains new users. That's why we do this operation).

3) Model Training: Using $train\_test\_split$ from *sklearn*, we split them into training data(0.8) and testing data(0.2). Feeding them into *lightGBM* and *XGB*, we find that *XGB* needs more computation time and there is a time limit in this competition. Therefore, we use *lightGBM* instead of *XGB* (Which means we can sample more data to increase accuracy). The parameters is as follows:

- $objective$: $binary$
- $metric$: auc
- num_leaves: 1000
- max_depth: 10
- $feature\_fraction$: 0.75
- $bagging\_freq$: 10
- $bagging\_fraction$: 0.80
- $learning\_rate$: 0.05

Then, we get **0.745**.

B. Model 2

1) Data Preprocessing: Compared to *Model 1*, this data added the information of timestamp. The original data is very large, so we need to sample some of the entire dataset. Unlike model 1, this model uses the timestamp as one of the features. If we simply pick the data randomly, it will destroy the structure of the timestamp feature. In order to fix the problem, I decided to pick the data based on every different user_id, meanwhile maintaining the interval of each timestamp. In this model we use more features to train, the maximum size we used to train this model is **25000000 (around 25% of all data)**.

2) Data Preparation: Same with *Model 1*, this method also uses the correct rate of each user and question. Furthermore, this method also calculates the average elapsed_time for every user and question, and the average rate that if the question had been explained for every user and question. After doing such calculation, we get

- $user\_avg\_correctly$
- $user\_avg\_explanation$
- $user\_avg\_elapsed\_time$
- $question\_avg\_correctly$
- $question\_avg\_explanation$
- $question\_avg\_elapsed\_time$

this six features. Apart from this, in this model we generate four new features based on the

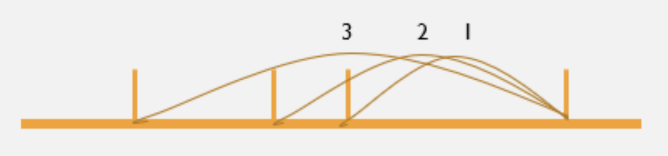

Figure 1. Illustration of the features we generated.

original timestamp data. The first feature is $user\_last\_incorrect\_timestamp$, which records the last timestamp that the user had a wrong answer. Other three features are $user\_timestamp\_1$, $user\_timestamp\_2$, $user\_timestamp\_3$, $user\_timestamp\_1$ is this timestamp minus the last one, $user\_timestamp\_2$ is this timestamp minus the last of the last one, same logic to generate $user\_timestamp\_3$. As shown as *Figure 1*.

3) Model Training: The training model of this method is *xLGBM*. The implement method is the same with *Model 1*. The score is **0.765**.

III. Experiment and Result

A. Model 1 (baseline)

For sampling size, we can find that the more we sample the higher score we get. Again, that's why we use *lightGBM* instead of *XGB* (Which means can we sample more data to increase accuracy).

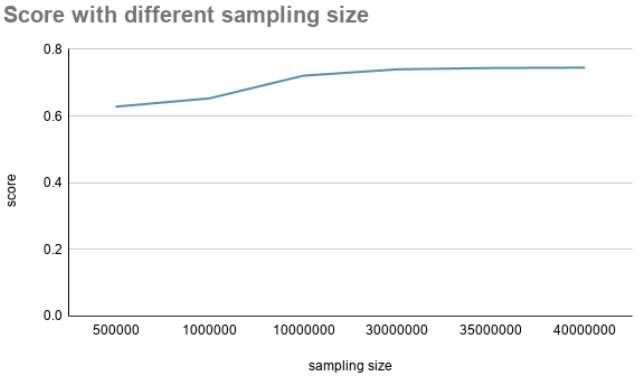

Figure 2. Score with Different Sampling Size

B. Model 2

There is a problem when we minus the timestamp to generate user timestamps 1, 2 and 3. The users were not answering the questions continuously, so the value of subtracting two consecutive timestamps sometimes tend to be very big, which means the users might do something else instead of answering questions. At first we thought that it's outliers, we replaced it by its average. Somehow, the accuracy drops to the same as model 1, which meant the timestamps interval we provided were useless. The reasonable explanation is that those outliers provide the information of who are not the loyal users that tend to do bad while answering questions.

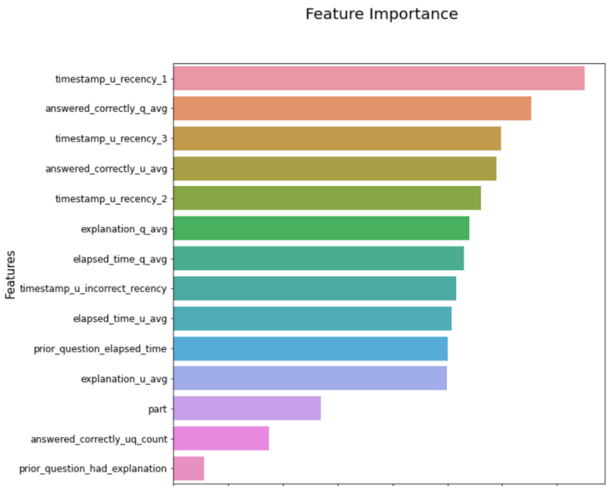

Figure 3. Feature Importance

Most of the timestamp features have higher importance than average rate information, and that is why model 2 performs better than model 1. The feature importance is shown as *Figure 3*.

IV. Conclusion

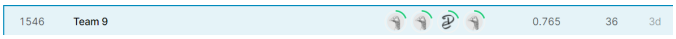

Figure 4. Public Leaderboard (Use for practicing)

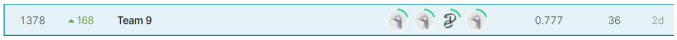

Figure 5. Private Leaderboard (The final score for the competition)

Figure 5 is our final score in the private leaderboard. Compared to the public leaderboard, the ranking increased by 168, the score also slightly improved. Which means that the robustness of our model is very strong, even using the totally different and very large data still remains good performance.