KLE Society's

KLE Technological University

A Course Project On

Riiid AIEd Challenge 2020 (5DMACP03)

submitted for the course of

Data Mining and Analysis (18ECSC301)

Submitted by

Team D1

Sumit Kumar Singh	01FE18BCS231
Sneha Majjigudda	01FE18BCS211
Swati S Mugda	01FE18BCS237
Naveen Kumar	01FE19BCS426

School of Computer Science and Engineering

Academic Year 2020-21.

Table of Contents

1. Introduction

For many educators, personalised learning is about using adaptive technology that can adjust and adapt to a student's skill level. Personalized learning as a comprehensive implementation can be the way nurture each individual student's social, emotional, and physical development. The idea is to customize the learning experience for each student according to his/her skills, abilities, past performance and experiences.
Online tutoring systems have been using the well known knowledge tracing model, popularized by Corbett and Anderson (1995), to track student knowledge for over a period of time. Surprisingly, models currently in use do not allow for individual learning rates nor individualized estimates of student initial knowledge.

Knowledge tracing(KT) is the task of modelling student knowledge over time so that we can accurately predict how students will perform on future interactions. Improvement on this task means that resources can be suggested to students based on their individual needs, and content which is predicted to be too easy or too hard can be skipped or delayed.Machine learning solutions could provide these benefits of high quality personalized teaching to anyone in the world for free. The knowledge tracing problem is inherently difficult as human learning is grounded in the complexity of both the human brain and human knowledge.

We have seen how a pandemic and other factors can drive institutionalised activities like learning to unknown paradigms of online education. To maintain the sanctity of learning and to better understand how we can disseminate not just information but more impactful information is the need of the hour.Machine learning and data mining techniques provide us expertise in working with massive data of students' performance. Data mining is a step in the overall Knowledge Discovery with Database process that contains preprocessing, data mining and post processing.

2. Problem Statement

In Riiid AIEd Challenge 2020, the challenge is to create algorithms for "Knowledge Tracing," the modeling of student knowledge over time. The goal is to accurately predict how students will perform on future interactions.

Riiid Labs, an AI solutions provider delivering creative disruption to the education market, empowers global education players to rethink traditional ways of learning leveraging AI. In 2020, the company released EdNet, the world's largest open database for AI education containing more than 100 million student interactions. With the use of EdNet, the newer Knowledge Tracing algorithms created can be equipped by learning systems post COVID-19 world to better personalized education.

The data provided by the challenge is about the students that are preparing for the TOEIC test. The TOEIC L&R uses an optically-scanned answer sheet. There are 200 questions to answer in two hours in Listening (approximately 45 minutes, 100 questions) and Reading (75 minutes, 100 questions). The same question format is used each time. Mark your answers on your answer sheet and not in the test book. The test is only in English. There are no questions involving translating from Japanese to English, or from English to Japanese.

3. Literature Survey

In this section, the summary of papers and/or publications referred before deciding upon how to go about solving the problem at hand.

3.1 EdNet: A Large-Scale Hierarchical Dataset in Education [1]

EdNet is a dataset consisting of all student-system interactions collected over a period spanning two years by Santa, a multi-platform AI tutoring service with approximately 780,000 students in South Korea. Santa is available through Android, iOS and the Web. It aims to prepare students for the TOEIC (Test of English for International Communication R) Listening and Reading Test.

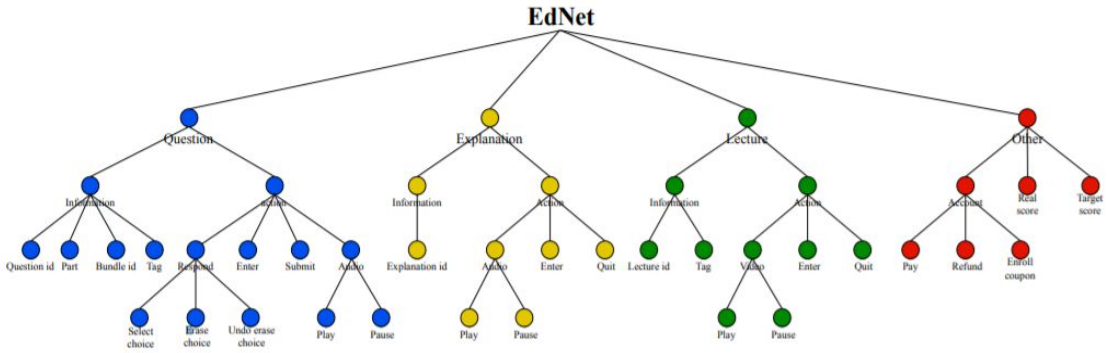


Figure 1: Hierarchical structure of EdNet

The raw records obtained by Santa accurately and thoroughly represent each student's learning process. However, the unprocessed details of raw records are hard to utilize directly for AIEd tasks and require pre-processing to extract meaningful information. In order to aid the process, we pre-process the collected records into four datasets of different levels of abstractions named KT1, KT2, KT3 and KT4. The resolution of each dataset increases in the given order, starting from question-response interaction sequences used by most deep knowledge tracing models to the complete list of user actions gathered by Santa.

3.2 Deep Knowledge Tracing [2]

The Recurrent Neural Networks (RNN) family of models have important advantages in that they do not require the explicit encoding of human domain knowledge, and can capture more complex representations of student knowledge. Using neural networks results in substantial improvements in prediction performance on a range of knowledge tracing datasets.

Deep Knowledge Tracing (DKT) method applies flexible RNNs that are 'deep' in time to the task of knowledge tracing. This family of models represents latent knowledge state, along with its temporal dynamics, using large vectors of artificial 'neurons', and allows the latent variable representation of student knowledge to be learned from data rather than hard-coded,whereas it fails in not generalizing well while dealing with sparse data which is the case with real world data as students interact with few knowledge concepts.

3.3 A Self-Attentive model for Knowledge Tracing [3]

KT is the task of modeling each student's mastery of knowledge concepts (KCs) as (s)he engages with a sequence of learning activitiesIn the KT task, the skills that a student builds while going through the sequence of learning activities, are related to each other and the performance on a particular exercise is dependent on his performance on the past exercises related to that exercise.A self-attention based knowledge tracing model(SAKT), it models a student's interaction history (without using any RNN) and predicts his performance on the next exercise by considering the relevant exercises from his past interactions.

It handles the data sparsity problem better than the methods based on RNNs,but it fails to leverage deep self-attentive computations for knowledge tracing. As a result, they fail to capture complex relations among exercises and responses over time. Secondly, appropriate features for constructing queries, keys and values for the self-attention layer for knowledge tracing have not been extensively explored.

3.4 LightGBM: A Highly Efficient Gradient Boosting Decision Tree [4]

Gradient Boosting Decision Tree (GBDT) is a popular machine learning algorithm.GBDT is an ensemble model of decision trees, which are trained in sequence.In each iteration, GBDT learns the decision trees by fitting the negative gradients.Gradient-based One-Side Sampling (GOSS),While there is no native weight for data instance in GBDT, we notice that data instances with different gradients play different roles in the computation of information gain .Exclusive

Feature Bundling (EFB), in a sparse feature space, many features are (almost) exclusive, i.e., they rarely take nonzero values simultaneously.

LightGBM, which contains two novel techniques: Gradient-based One-Side Sampling and Exclusive Feature Bundling to deal with large numbers of data instances and large number of features respectively.It is more efficient than other gradient boosting algorithms as it reduces the optimal bundling problem to a graph coloring problem, and solving it by a greedy algorithm with a constant approximation ratio.

3.5 Towards an Appropriate Query, Key, and Value Computation for Knowledge Tracing [5]

It is a Transformer-based model for knowledge tracing, SAINT: Separated Self-Attentive Neural Knowledge Tracing. SAINT has an encoder-decoder structure where the exercise and response embedding sequences separately enter, respectively, the encoder and the decoder.The encoder applies self-attention layers to the sequence of exercise embeddings, and the decoder alternately applies self attention layers and encoder-decoder attention layers to the sequence of response embeddings. This separation of input allows to stack attention layers multiple times, resulting in an improvement in area under receiver operating characteristic curve (AUC).Separating the exercise sequence and the response sequence, and feeding them to the encoder and decoder,respectively, are distinctive features that allow SAINT to capture complex relations among exercises and responses through deep self-attentive computations.

4. Methodology

The methodology adopted to implement the knowledge tracing algorithm is KDD. Knowledge Discovery in Database (KDD) process in data mining is a programmed and analytical approach to model data from a database to extract useful and applicable 'knowledge'. It utilises algorithms that are self-learning in nature to deduce useful patterns from the processed data.

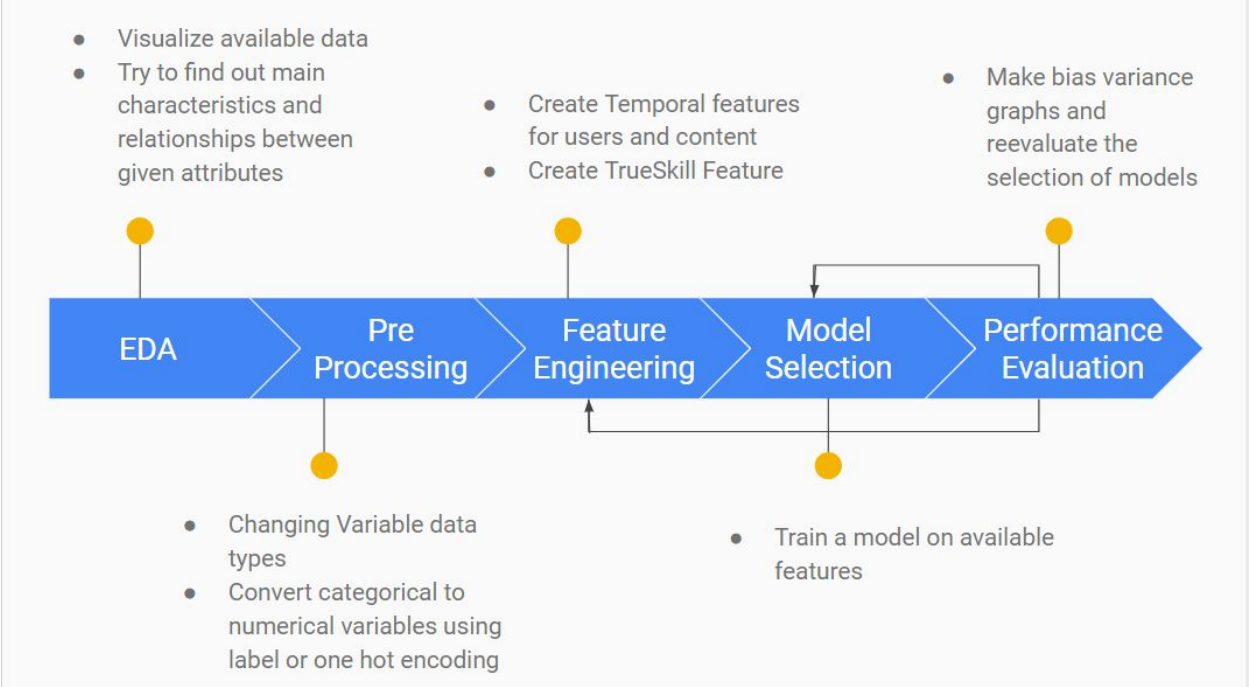


Figure 2: Iterative KDD process

4.1 Pre-processing

To predict whether students will be able to answer the next question correctly, data regarding the student's historic performance along with metadata for questions and lectures is provided.

Data pre-processing includes data cleaning, transformation and reduction. Data cleaning incorporates handling the missing quantities and removal of noise or outliers to make the data more reliable. Data cleaning practices like dropping any unwanted, redundant attributes and filling in missing values. Before any attributes are termed as irrelevant, correlation with the other

attributes has to be verified so that removal of the attribute does not disturb the consistency of the data. The missing values in the dataset are handled by 'highlighting' them. That is, inserting a value like 0, to inform the model that the data is missing.

Data transformation is a technique used to convert the raw data into a suitable format that eases data mining in retrieving the strategic information efficiently. Converting the data types of the attributes is one of the data transformation techniques employed. This technique also helps in memory optimization, which is a factor that affects the model(s), as more memory consuming data types like 'int64' can be converted to lesser memory consuming data types like 'int8' or ' int16' depending on the attributes.

In data reduction, techniques like decimal scaling can be adopted to scale down the values of attributes exponentially. This can help in faster and efficient computations using those attributes.

4.2 Feature Engineering

Feature Engineering of "temporal" features, that change over time for a particular user, proves to be beneficial while trying to do knowledge tracing. We even used Microsoft TrueSkill™ Ranking System that is a ranking system used in XBOX for matchmaking teams on games. We modelled it in a way where it treats a student as one team and a question as another. Depending on how a student performs on a easy or difficult question (a lower trueskill quotient indicates that the question is easy) the level of the student is set and performance on future questions is predicted.

After preprocessing the available data, feature engineering of it gave us features that could be used to make better predictions on our problem statement. **User_interaction_count** and **User_interaction_timestamp_mean** are engineered features that tell the number of content ids a user interacts with and the average time the student takes to complete these interactions and move on to the next interactions. It is obtained by using the given timestamp attribute and calculating the mean of it for each particular user. **Attempt_no** is an engineered feature that gives us a count of the number of times a particular content has been integrated with by a particular user.

Lagtime is an engineered feature that tells us the time a user takes between successive interactions. **Lagtime2** and **Lagtime3** are similar features derived from the given timestamp attribute. These features tell us about the time taken by a user between 3 and 4 successive questions respectively.

User_lecture_sum is a feature that is calculated by segregating the lecture content ids and question content ids per user and taking the cumulative sum of those interactions. **User_lecture_lv** uses user_lecture_sum to give the overall percentage of lecture interactions per user on the total number interactions.

User_correct_count and **User_uncorrect_count** is the number of questions a particular user has answered correctly or has not answered correctly respectively. **User_correctness** is another engineered feature that is the ratio of the user_correct_count to the total number of interactions a particular user has had. **Content_correct_count** and **Content_uncorrect_count** is the number of times a particular content has been answered correctly and has been answered incorrectly. **Content_correctness** for a content id is the ratio between content_correct_count and the total number of times that content id appears in the entirety of the dataset.

Content_elapsed_time is a feature derived from the given elapsed_time attribute. This tells us the average time it takes for a particular question to be answered irrespective of the user. **Content_explanation_false_mean** and **Content_explanation_true_mean** are engineered features that tell us about the performance of a user on questions that were asked after the explanations to the previous bundle were not given and given respectively.

Correct_probability is a feature that is derived from the TrueSkill library that implements Microsoft TrueSkill Ranking System[6]. The TrueSkill ranking system is used by Microsoft in matchmaking services for games on its gaming console, XBOX. The TrueSkill System assigns an initial competence rating to each team. Here, we treat every single user as a team and every content id as a team as well. How a user performs against a question impacts the competence rating of both, the user as well as the question. The lesser the competence rating of a question the easier it is for a user to answer that question and the more is the competence rating of a question the harder it is to answer.

4.3 Model Selection

Boosting is one of the techniques that uses the concept of ensemble learning. It is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones. LGBM is a gradient boosting framework that uses tree based learning algorithms.

SAKT (Self Attentive Knowledge Tracing) outperforms other traditional Knowledge Tracing algorithms as it can handle the sparse data which is the case in most real world scenarios.The

main component (self-attention) of SAKT is suitable for parallelism. It identifies the KCs from a student's past activities that are relevant to the given KC and predicts his/her mastery based on the relatively few KCs that it picked. Since predictions are made based on relatively few past activities, it handles the data sparsity problem[9].

An ensemble of LGBM and SAKT was used to better predict the output. An ensemble model aggregates the prediction of each base model and results in one final prediction for the unseen.

4.4 Model Evaluation

Model evaluation aims at estimating the generalization error of the model(s), i.e., how well the selected model(s) perform on unseen data. The parameter used for evaluating the performance of the model is Area under the ROC curve (AUC). An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters namely, True Positive Rate and False Positive Rate.

$$True\ Positive\ Rate\ =\ Specificity\ = True\ Positives\ /\ (True\ Positives\ +\ False\ Negatives)$$

$$False\ Positive\ Rate\ =\ 1\ -\ Specificity\ = False\ Positives\ /\ (False\ Positives\ +\ True\ Negatives)$$

AUC stands for Area under the curve. AUC gives the rate of successful classification by the logistic model. The AUC makes it easy to compare the ROC curve of one model to another.

The performance of learning models on both training data and validation data is illustrated in learning curves. In a learning curve, the performance of a model both on the training and validation set is plotted as a function of the training set size. Learning Curves also help ascertain whether a model is underfitting, overfitting or generalizing well for any new data that comes into the system.

5. Results and Discussions

In the results section, the outcomes of the EDA, and the entire KDD process are discussed. Performance of the models are evaluated using the Model Evaluation parameters discussed in section 4.4.

5.1 Exploratory Data Analysis

A correlation heatmap shows the correlation between the attributes given in the training dataset. As shown in figure 3, the attributes do not show significant correlation. Analysing the correlation ultimately helps to decide which features affect the target variables the most, and in turn, which features are used in predicting the target variable value.

Figure 3 : HeatMap of Correlation Matrix

The attribute prior_question_had_explanation is of the type boolean and indicates whether or not the user saw an explanation and the correct response(s) after answering the previous question bundle, ignoring any lectures in between. The value is shared across a single question bundle,

and is null for a user's first question bundle or lecture. The attribute answered_correctly indicates if a student has answered a question correctly or not.

The graph in figure 4 shows that if a student has seen the explanation of the question in the previous bundle, he/she is more likely to answer the questions in the current bundle correctly. Hence, the attribute prior_question_had_explanation seems to contribute to the value of the target attribute.

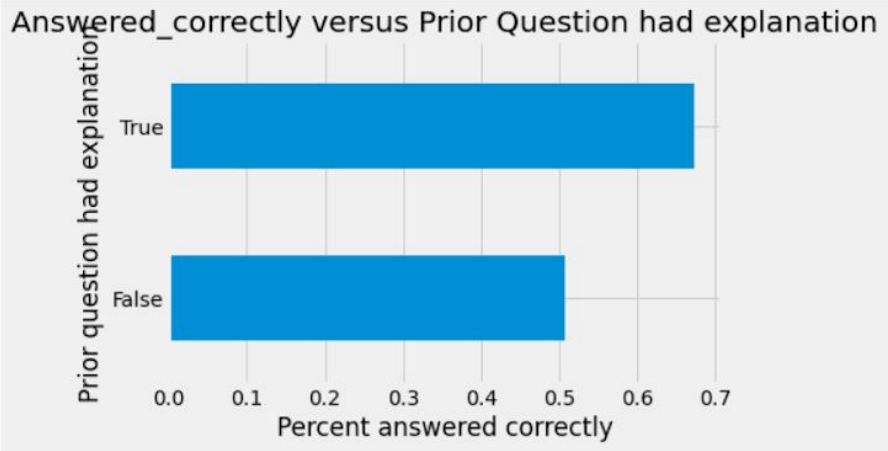


Figure 4: Answered Correctly vs Prior Question has Explanation

5.2 Performance Evaluation

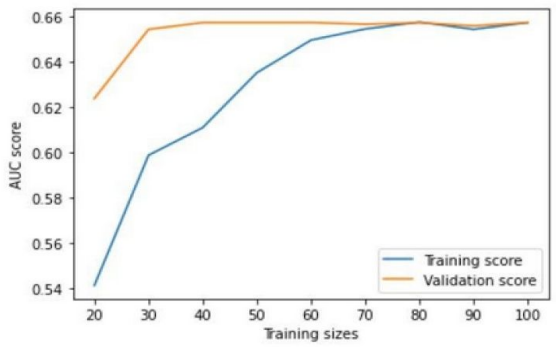
In this section, the performance of all the models built during the KDD iterations. Table 1 shows the AUC score (evaluation metric) of how the models fared on the test dataset.

Learning Model	Best AUC score obtained
Decision Tree Classifier	**0.5968**
AdaBoost	0.6760
Logistic Regression	0.6570
LGBM	**0.7568**
Random Forest	0.6640

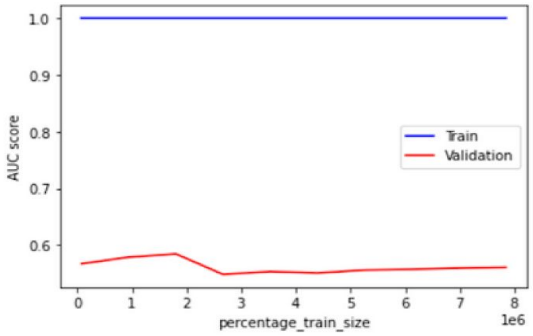
K-Nearest Neighbour Classifier	0.6121
SVM	0.6720
Naive Bayes Classifier	0.6581
CatBoost	0.7489
LGBM + SAKT	**0.785**

Table 1: Results of learning models

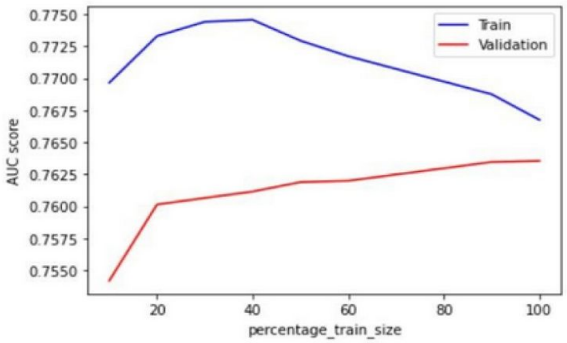
Learning curves show the relationship between training set size and the chosen evaluation metric (e.g. AUC, accuracy, etc.) on the training and validation sets. Learning curves are used to diagnose model performance.



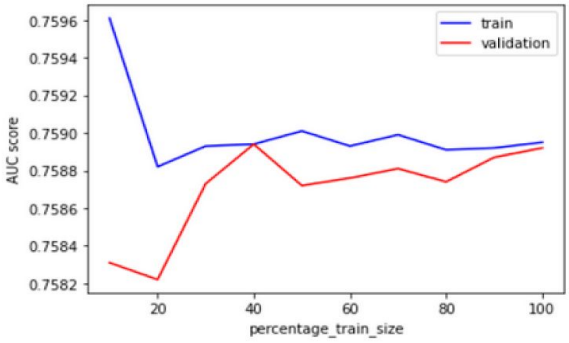
(a) Logistic Regression



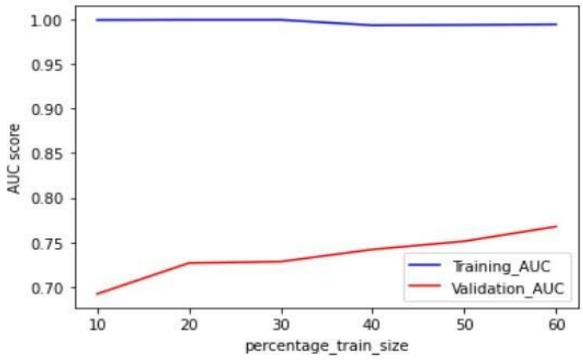
(b) Decision Tree



(c) LGBM



(d) CatBoost



(e) Random Forest

Figure 5: Learning Curves of models built

Fig 5(a) shows the learning curve for logistic regression which shows slight underfitting with low AUC score for training data compared to validation data for a logistic regression model. The training AUC score for decision tree 1.00 regardless the size of dataset whereas the validation score is less than 0.6 showing clear overfitting as seen in Fig 5(b).
With the increase in size of data, the learning curves for the LGBM and CatBoost in Fig 5(c) and Fig 5(d) respectively, show best-fit. The difference between training data and validation data is insignificant and decreases with increasing data size.
The learning curve for the Random forest model shows overfitting in Fig 5(e) with significant difference between training and validation data. The training AUC score is nearly 1.00 while the model fails to validate it for new data.

After the performance evaluation, the model with the best AUC score and appropriate fit, LGBM is chosen. LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with faster training speed and higher efficiency, lower memory usage, better accuracy, support of parallel and GPU learning and capable of handling large-scale data.

The features related to ‘timestamp’ like ‘lagtime’ and ‘lagtime_mean’ improved the public score from 0.723 to 0.752. The correctness features ‘user_part_correctness’, ‘task_conatiner_correctness’, ‘tags_correctness’ along with ‘attemp_no’ raised the score to 0.768. Further, the addition of the ‘correct_probability’ derived from the TrueSkill library feature increased the score to 0.777.

Although additional feature engineering for the LGBM would increase the score, exploration of newer, DL-based knowledge tracing models was necessary. SAKT (Self Attentive

Knowledge Tracing) model is one which utilizes Transformer's self attention architecture. It is a single-encoder based model where exercise ids are used as queries and interaction ids are used as keys and values. For effective knowledge tracing, one needs to analyze the relationships between the different parts of learning activity data. The attention mechanism in SAKT serves that purpose by weighing the more relevant parts of data heavier for prediction. The inclusion of SAKT to the existing LGBM model raised the score to 0.785.

5.3 Post Processing

In post-processing, we revisit the earlier stages of the KDD process- pre-processing, feature engineering and improve the performance of the chosen model.

In the first iteration of KDD process, performance of the student as 'user_correctness', total number of questions attempted by each student as 'content_count' and the number of questions answered correctly as 'content_sum' are some features that are extracted from aggregation of 'user_id' and 'content_id' in train.csv. Further, the number of times a question is correctly answered by any student can be found by dividing 'content_sum' by 'content_count'. This is added as a feature called 'content_correctness'.

The 'tags' given in questions.csv can be clustered into different communities which represent the difficulty of questions. Hence 'community' is modelled as a feature.

Parameter tuning of the model also contributes to better performance. Parameters like 'early_stopping_rounds' according to which if the accuracy of the model does not improve for a set number of iterations then the model training would be stopped, 'learning_rate' which is the parameter that adjusts the loss function for any model, 'num_iterations', 'num_leaves' which controls the complexity of an LGBM model are tuned to get better accuracy for the model Decreasing num_leaves reduced the training time however it was more beneficial to increase the num_leaves as it increases the accuracy of the model .

Features like 'lagtime' and 'lagtime_mean' for each user that describe the time each user spends between interactions are added. These values update with every new user interaction that the model encounters i.e., they are temporal. This ensures that the model is not working with information that might be outdated. A feature 'user_part_correctness' that describes for each user the average correctness per part. A user with a high 'user_part_correctness' value for a specific part is more likely to answer the questions that appear in that TOEIC part correctly. Similarly,

features like ‘task_container_correctness’,‘tags_corectness’ and ‘attempt_no’ also increase the performance.

A notable feature is the ‘correct_probability’ of a user, derived from the TrueSkill library feature. Every user and unique content_id are treated as participants. Every time a user answers a question correctly, the rating for the user increases and the rating for the content_id decreases. Viceversa if the user answers a question incorrectly. To better represent the dataset, 30 most recent interactions of each user are chosen and used for training to the model.

6. Conclusion

The question that Riiid! asks is whether one can determine a student's answer correctness on a question asked based on past performances. If so, how and how effectively? The problem is that of a binary classification nature. The output can either be that a student answers a question correctly or not. This prompts the usage of Classifier algorithms like Decision Trees and Support Vector Machines. Gradient Boosting Algorithms that produce an ensemble of weaker classification algorithms like decision trees performed well in making predictions. LGBM proved to be the best classifier for the given problem statement. It creates a big pool of decision trees and picks and chooses the best attributes and parameters. Tuning the parameters of an LGBM classifier can improve its prediction capabilities as well. An ensemble of LGBM and SAKT performed best as it gave the maximum AUC score. Self Attention Transfer Models use deep learning techniques which make it more efficient in making predictions.

This challenge turned out to be one of feature generation and parameter tuning of classifier algorithms. Altering the number of leaves so as to avoid overfitting, altering the maximum depth of the classifier so as to make predictions quicker. The tuning of parameters of the LGBM classifier was an important learning of the challenge as well. In addition to user based features like ‘user_correctness’, ‘lagtime’ and ‘user_part_target_aggregations’ - a measure of how a student performs on a question based on which part of the TOEIC test appears; showed improvement in prediction capabilities. Such temporal features add great value in knowledge tracing problems. Temporal Features are features that change for a particular user that change over time. The more such features we have the better the capability of the model would be to predict the outcome.

Another major learning from the challenge was that of memory optimization when working with large data sets like that of EdNet. Algorithms like LGBM and Random Forest Classifier that take up a huge amount of primary memory when making trees needed to be used efficiently as Kaggle provides a small limited amount of primary memory.

7. References

[1] Choi, Youngduck, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Chan Bae, Byungsoo Kim, and Jaewe Heo. "Ednet: A large-scale hierarchical dataset in education." In *International Conference on Artificial Intelligence in Education*, pp. 69-73. Springer, Cham, 2020.

[2] Piech, Chris, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. "Deep knowledge tracing." *arXiv preprint arXiv:1506.05908* (2015).

[3] Pandey, Shalini, and George Karypis. "A self-attentive model for knowledge tracing." *arXiv preprint arXiv:1907.06837* (2019).

[4] Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems* 30 (2017): 3146-3154.

[5] Choi, Youngduck, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. "Towards an appropriate query, key, and value computation for knowledge tracing." In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pp. 341-344. 2020.

[6] TrueSkill Ranking System by Microsoft