Student Knowledge Tracking and Answer Prediction System.

1st Ankur Chaudhari
Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey, United States
achaud6@stevens.edu

2nd Sudipto Sen
Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey, United States
ssen2@stevens.edu

3rd Ali Tootoonchi
Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey, United States
atootoon6@stevens.edu

Abstract—**Think back to your favorite teacher, They motivated and inspired you to learn. They knew your strengths and weaknesses and modified their teaching style based on your ability. For example, the teacher would make sure you understood algebra before advancing to calculus. In 2018, 260 million students weren't attending school and presently "Education" is in a tough place since COVID-19 has forced many countries to temporary close down schools. The aim of this project is to provide access to personalize training for students who need it the most. We utilize data released by EdNet [3], the world's largest open database for AI education, containing more than 100 million student interactions. We are creating Machine Learning models for "Knowledge tracing", the modelling of student knowledge over time using Time-series forecasting.**

Index Terms—**Machine Learning, time-series forecasting, EdNet, AI**

I. Introduction

With advances in Artificial Intelligence in Education (AIEd) and the ever-growing scale of Interactive Educational Systems (IESs), data-driven approach has become a common recipe for various tasks such as knowledge tracing and learning path recommendation.

Although several datasets, such as ASSISTments, Junyi Academy, Synthetic and STATICS, are publicly available and widely used, they are not large enough to leverage the full potential of state-of-the-art data-driven models and limits the recorded behaviors to question-solving activities. To this end, we utilize EdNet, a large-scale hierarchical dataset of diverse student activities collected by Santa, a multi-platform self-study solution equipped with Artificial Intelligence tutoring system. EdNet contains 131,441,538 interactions from 784,309 students collected over more than 2 years, which is the largest among the ITS [4] datasets released.

After importing this dataset, we utilize it to closely analyse students' interactions to gain a deeper understanding of the distribution of individual variables as well as the relationships between independent and dependent variables. We want to create meaningful features which will help us in creating a good model which is well tuned to the data. Finally the model is implemented after preprocessing the data as discussed and observations towards answer predictions are recorded. Thus, the student knowledge is modelled over time using Time-series forecasting using three different Machine Learning algorithms.

II. Problem Description

COVID-19 has brought upon us the inevitable transformation towards virtual education. The ensuing need for scalable, personalized learning systems has led to an unprecedented demand for understanding large-scale educational data. However, the field of Artificial Intelligence in Education (AIEd) has received relatively little academic attention compared to the mainstream machine learning areas like vision, natural language processing, and healthcare.

III. Dataset Description

Tailoring education to a student's ability level is one of the many valuable things an AI tutor [1] can do. We intend predict whether students are able to answer their next questions correctly. We are provided with the same sorts of information a complete education app would have: that student's historic performance, the performance of other students on the same question, metadata about the question itself, and more.

A. Training Dataset

Train.csv file contains 4.5GB of raw students interactions data containing 10 attributes focused on student's academic records.

- row_id : ID code for the row.
- timestamp :the time in milliseconds between this user interaction and the first event completion from that user..
- user_id : ID code for the user.
- content_id :ID code for the user interaction.
- content_type_id : 0 if the event was a question being posed to the user, 1 if the event was the user watching a lecture..
- task_id : ID code for the batch of questions or lectures. For example, a user might see three questions in a row before seeing the explanations for any of them. Those three would all share a task_id.
- user_answer :(int8) the user's answer to the question, if any. Read -1 as null, for lectures.
- answered_corecorrectly :if the user responded correctly. Read -1 as null, for lectures.
- prior_question_elapsed_time :The average time in milliseconds it took a user to answer each question in the previous question bundle, ignoring any lectures in

between. Is null for a user's first question bundle or lecture. Note that the time is the average time a user took to solve each question in the previous bundle.
- prior_question_had_explanation : 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. Typically the first several questions a user sees were part of an onboarding diagnostic test where they did not get any feedback.

B. Questions Dataset

questions.csv file contains 288.22 of questions records asked to students as well as information about the topics to which questions are related.

- question_id : foreign key for the train/test content column, when the content type is question (0).
- bundle_id : code for which questions are served together.
- correct_answer : the answer to the question. Can be compared with the train user_answer : column to check if the user was right.
- part : the relevant section of the TOEIC test [5].
- tags : one or more detailed tag codes for the question. The meaning of the tags will not be provided, but these codes are sufficient for clustering the questions together.

C. Lectures Dataset

lectures.csv file contains 9.48 of lectures records and topics covered.

- lecture_id : foreign key for the train/test contentcolumn, when the content type is question (0).
- type_of : brief description of the core purpose of the lecture.
- part : top level category code for the lecture.
- tag : one tag codes for the lecture. The meaning of the tags will not be provided, but these codes are sufficient for clustering the lectures together.

IV. Tools and Libraries

RAPIDS [6] is a suite of open-source software libraries and APIs for executing data science pipelines entirely on GPUs. Some of the advantages are -

A. Overview of RAPIDS

RAPIDS is a suite of open-source software libraries and APIs for executing data science pipelines entirely on GPUs. Some of the advantages are -

- Faster Execution Time - RAPIDS leverages NVIDIA CUDA® under the hood to accelerate workflows by running the entire data science training pipeline on GPUs. This reduces training time and the frequency of model deployment from days to minutes.
- Use the Same Tools - By hiding the complexities of working with the GPU and even the behind-the-scenes communication protocols within the data center architecture, RAPIDS creates a simple way to get data science done.
- Use the Same Tools - By hiding the complexities of working with the GPU and even the behind-the-scenes communication protocols within the data center architecture, RAPIDS creates a simple way to get data science done.

We initially started working on this problem in Pandas, but the code compilation was quite slow and we weren't able to handle the datasets properly. Therefore we switched to RAPIDS and since then the overall runtime of the notebook has come down to 2 mins(excluding load time for CuDf packages)

B. CuML

cuML is a suite of libraries that implement machine learning algorithms and mathematical primitives functions that share compatible APIs with other RAPIDS

C. CuDf

CuDF offers a pandas-like API that data engineers and data scientists would be familiar with, so that they can use it to speed up their workflows without going into the specifics of CUDA programming.

D. CuPy

CuPy is a CUDA implementation of the NumPy-compatible multi-dimensional array. CuPy consists of the main multi-dimensional array class, cupy.ndarray, and a variety of functions on it.

V. Preprocessing

A. Preprocessing on train.csv

- Split train.csv into train and validation dataset : The use of the last few entries for each user as validation data is simple and doesn't look too bad. However this split-method can concentrate too much on light users over heavy users. As a result, the average percentage of correct answers is lower, and there could be a chance of leading us in the wrong direction.
- Removing data with content_type_id == 1 : As content_type_id with value 1 contains data as event as student was watching a lecture.
- Since the training data and the test data are separated by time, the validation data should also be segregated by time. However the time stamp is the time that has passed after the first activity of the user, not the actual time. So we set a random first access time for each user within a certain period of time.

B. Adding Features to dataset

- answered_correctly_sum and answered_correctly_average for every student is calculated to generate performance feature for every student.

- Training and validation data is merged with calculated data in key content_id.
- Null values for attribute prior_question_elasped_time is replaced with mean values.

C. Preprocessing on questions.csv

- Null values for attribute prior_question_had_explanation with is replaced value False.
- questions.csv is merged file with train and validation data as left joint on left attribute as content_id and right attribute question_id

VI. Algorithms

In this project we plan to implement three Machine Learning algorithms and observe their predictions in order to select the best model as well as compare their accuracy. The three models proposed are - LightGBM [7], XGBoosting and SVM.

A. LightGBM

LightGBM, short for Light Gradient Boosting Machine, is a free and open source distributed gradient boosting framework for machine learning originally developed by Microsoft. It is based on decision tree algorithms and used for ranking, classification and other machine learning tasks. The development focus is on performance and scalability.

The LightGBM framework supports different algorithms including GBT, GBDT, GBRT, GBM, MART and RF. LightGBM has many of XGBoost's advantages, including sparse optimization, parallel training, multiple loss functions, regularization, bagging, and early stopping. A major difference between the two lies in the construction of trees. LightGBM does not grow a tree level-wise — row by row — as most other implementations do. Instead it grows trees leaf-wise. It chooses the leaf it believes will yield the largest decrease in loss.

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency: Light GBM use histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.
- Lower memory usage: Replaces continuous values to discrete bins which result in lower memory usage.
- Better accuracy than any other boosting algorithm: It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter
- Compatibility with Large Datasets: It is capable of performing equally good with large datasets with a significant reduction in training time as compared to XGBOOST.
- Capable of handling large-scale data.

Tuning Parameters of Light GBM Light GBM uses leaf wise splitting over depth-wise splitting which enables it to converge much faster but also leads to overfitting. Following section states paramter tuning effect on efficiency of LightGBM model.

For best fit

- num_leaves : This parameter is used to set the number of leaves to be formed in a tree. Theoretically relation between num_leaves and max_depth is num_leaves= $2\hat{(}$max_depth). However, this is not a good estimate in case of Light GBM since splitting takes place leaf wise rather than depth wise. Hence num_leaves set must be smaller than $2\hat{(}$max_depth) otherwise it may lead to overfitting. Light GBM does not have a direct relation between num_leaves and max_depth and hence the two must not be linked with each other.
- min_data_in_leaf : It is also one of the important parameters in dealing with overfitting. Setting its value smaller may cause overfitting and hence must be set accordingly. Its value should be hundreds to thousands of large datasets.
- max_depth: It specifies the maximum depth or level up to which tree can grow.

For faster speed

- bagging_fraction : Is used to perform bagging for faster results feature_fraction : Set fraction of the features to be used at each iteration
- max_bin : Smaller value of max_bin can save much time as it buckets the feature values in discrete bins which is computationally inexpensive.

For better accuracy

- Use bigger training data
- num_leaves : Setting it to high value produces deeper trees with increased accuracy but lead to overfitting. Hence its higher value is not preferred.
- max_bin : Setting it to high values has similar effect as caused by increasing value of num_leaves and also slower our training procedure.

B. XGBoost

XGBoost [8] stands for eXtreme Gradient Boosting. XGBoost is an open-source software library which provides a gradient boosting. It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. It belongs to a broader collection of tools under the umbrella of the Distributed Machine Learning Community or DMLC who are also the creators of the popular mxnet deep learning library. It works on Linux, Windows,[7] and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library".

- Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of different prediction models.

- It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

For a fixed set of hyper-parameters, XGBoost provides the largest reduction in training time when using a GPU relative to a CPU. For the XGBoost implementation in this project, the training time has reduced from over 35 mins to 3.34 secs! Moreover, we observe that one is often able to utilize this speedup to converge to a good set of hyper-parameters in a short time.

The following points emphasise the advantages of XGBoost algorithim.

- Regularization: XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XGBoost is also called regularized form of GBM (Gradient Boosting Machine).While using Scikit Learn libarary, we pass two hyper-parameters (alpha and lambda) to XGBoost related to regularization. alpha is used for L1 regularization and lambda is used for L2 regularization.
- Parallel Processing: XGBoost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model. While using Scikit Learn libarary, nthread hyper-parameter is used for parallel processing. nthread represents number of CPU cores to be used.
- Effective Tree Pruning: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XGBoost on the other hand make splits upto the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.
- Handling Missing Values: XGBoost has an in-built capability to handle missing values. When XGBoost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

Just like Gradient Boost, XGBoost is the extreme version of it. When we use XGBoost, no matter we use it for classification or regression, it starts with an initial prediction and we use loss function to evaluate if the prediction works well or not.

The working of XGBoost model:

- At first, we put all residuals into one leaf and calculate the similarity score by simply setting lambda =0 . Then we consider whether we could do a better job clustering similar residuals if we split them into 2 groups. Then we calculate the similarity for each groups (leaf and right). Then we quantify how much better the leaves cluster similar residuals than the root by calculating the gain.
- Gain = Left similarity + Right similarity- Root similarity
- After this, we could compare the gain with this and gain with other thresholds to find the biggest one for better split.
- For the leaves could be split, we continue the splitting and calculate the similarity score and gain just as before. Then we will talk about tree pruning based on its gain value. We start by picking a number as threshold, which is gamma. Then we calculate the difference between the gain associated with the lowest branch in the tree and the value for gamma (Gain-gamma). If the difference between the gain and gamma is negative, we remove the branch. If it is positive we will keep the branch so we finish the pruning. Note, we will never remove the root if we do not remove the first branch. However, if we prune the root, it shows us the initial prediction is all we left which is an extreme pruning.
- Then we go back to the original residuals and build a tree just like before, the only difference is we change the lambda value to 1. We calculate the similarity score and gain in just the same way and we found that when lambda is larger than 0, the similarity and gain will be smaller and it is easier to prune leaves. The lambda prevented over-fitting the training data.
- After we build the tree, we start to determine the output value of the tree. Then we can make prediction based on the tree. The new prediction is:
- Initial predicted value + learning rate (eta) x output value
- We could simply compare the new residuals and found that whether we have taken a small step in the right direction. We keep building other trees based on new residuals and make new prediction gives smaller residuals until residuals are supper small or reached maximum number.

C. *CatBoost*

C-Support Vector Classification.

The implementation is based on libsvm [9]. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using sklearn.svm.LinearSVC or sklearn.linear_model.SGDClassifier instead, possibly after a sklearn.kernel_approximation.Nystroem transformer.

The multiclass support is handled according to a one-vs-one scheme.

VII. Observation

Results generated with default parameters for LightGBM, XGBoost, and Catboost model.

VIII. Environment

The API provides user interactions groups in the order in which they occurred. Each group will contain interactions from

TABLE I
MACHINE LEARNING MODEL RESULTS

Output Data	**Models**		
	LightGBM	***XGBoost***	***CatBoost***
Iterations	5000	5000	5000
AUC Score(Train)	0.762	0.7617	0.689
AUC Score(Validation)	0.762	0.7618	0.689
implementation Time	15min 8s	3.34s	10min 57s

many different users, but no more than one task_container_id of questions from any single user. Each group has between 1 and 1000 users. We expect to see roughly 2.5 million questions in the hidden test set.

The API will load up to 1 GB of the test set data in memory after initialization. The initialization step (env.iter_test()) will require meaningfully more memory than that; we must not load the model until after making that call. The API will also consume roughly 15 minutes of runtime for loading and serving the data, but will also obfuscate the true runtime for all submissions.

Custom Python module was created, ”riiideducation”. The purpose of this module is to control the flow of information to ensure that no future data is used to make predictions. Without proper usage of the module, the code may fail.

At present we are implementing the model by performing a train_test split on the 100 million entries provided. The environment plays an important role in determining the final accuracy of the model(s) implemented.

IX. INFERENCE

From current observations, we may conclude that for the data-set provided, XGBoost has the best performance, after sufficient tuning of the parameters, and provides a prediction within 3.5secs of training, for a 5 GB .csv data-set. It is worth mentioning that the model of LightGBM is a comparable in terms of accuracy, for the data-set provided but it takes longer (around 15 mins) for training. We can hope that for XGBoost accuracy will increase even further with finer tuning of the parameter. The performance of CatBoost has been satisfactory, however, accuracy is average and compilation is taking a lot longer for the training dataset.

We intend to select the model which has a perfect combination of high accuracy and low compilation time whilst avoiding over-fitting, in order to make predictions of students' answers based on the dataset describing their interactions. Hence, we shall choose XGBoost model for modelling student knowledge over time by prediction of answers using the most significant parameters. The data generated over time can prove to be very useful in the future as well, for it might help alleviate the current crisis in the education industry.

X. COMPARISON

A. Structural Differences in LightGBM XGBoost

LightGBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value while XGBoost uses pre-sorted algorithm Histogram-based algorithm for computing the best split. Here instances mean observations/samples.

First, let us understand how pre-sorting splitting works-

- For each node, we enumerate over all features
- For each feature, we sort the instances by feature value
- We use a linear scan to decide the best split along that feature basis information gain
- Finally, we take the best split solution along all the features

In simple terms, Histogram-based algorithm splits all the data points for a feature into discrete bins and uses these bins to find the split value of histogram. While, it is efficient than pre-sorted algorithm in training speed which enumerates all possible split points on the pre-sorted feature values, it is still behind GOSS in terms of speed.

Gradient represents the slope of the tangent of the loss function, so logically if gradient of data points are large in some sense, these points are important for finding the optimal split point as they have higher error

GOSS keeps all the instances with large gradients and performs random sampling on the instances with small gradients. For example, let's say I have 500K rows of data where 10k rows have higher gradients. So my algorithm will choose (10k rows of higher gradient+ x%of remaining 490k rows chosen randomly). Assuming x is 10%, total rows selected are 59k out of 500K on the basis of which split value if found.

The basic assumption taken here is that samples with training instances with small gradients have smaller training error and it is already well-trained. In order to keep the same data distribution, when computing the information gain, GOSS introduces a constant multiplier for the data instances with small gradients. Thus, GOSS achieves a good balance between reducing the number of data instances and keeping the accuracy for learned decision trees.

B. How each model treats Categorical Variables?

- CatBoost: CatBoost has the flexibility of giving indices of categorical columns so that it can be encoded as one-hot encoding using one_hot_max_size (Use one-hot encoding for all features with number of different values less than or equal to the given parameter value). If we don't pass any anything in cat_features argument, CatBoost will treat all the columns as numerical variables.
- LightGBM: Similar to CatBoost, LightGBM can also handle categorical features by taking the input of feature names. It does not convert to one-hot coding, and is much faster than one-hot coding. LGBM uses a special algorithm to find the split value of categorical features.
- XGBoost: Unlike CatBoost or LGBM, XGBoost cannot handle categorical features by itself, it only accepts numerical values similar to Random Forest. Therefore one has to perform various encodings like label encoding, mean encoding or one-hot encoding before supplying categorical data to XGBoost. However, since this project involves only prediction of answers, based on several

parameters, most of which does not depend on categorical features, it is no surprise that the XGBoost model has performed so well, with such a short training period.

XI. Future research directions

With a strong answer prediction system, we can create a personalized AI teacher for every student, that will not only guess whether a student will answer a question correctly in an examination, but also help the student by suggesting methods that will help the student maximise his score in examinations and therefore help improve his/her knowledge of a subject. With a strong personalized AI teacher that adapts to the student's needs in order to modify the course structure for maximising the student's output, we can alleviate all problems within the education industry. Several students who do not have access to proper schools, can benefit from the guidance of such an AI teacher.

XII. Conclusion

A. LightGBM

- On implementing the model with default parameters, we get the following output:
  Training's auc: 0.761653 valid_1's auc: 0.761832
  CPU times: user 28min 35s, sys: 12.9 s, total: 29min 16s
  Wall time: 15min 52s
- Whereas on fine-tuning the parameters, we get the following output:
  Training's auc: 0.763247 valid_1's auc: 0.763592
  CPU times: user 27min 50s, sys: 12.4 s, total: 28min 2s
  Wall time: 15min 8s

B. XGBoost

- On implementing the model with default parameters, we get the following output:
  XGBoost: ROC AUC = 74.982.
  CPU times: user 35min 41s, sys: 3.78 s, total: 35min 45s
  Wall time: 35min 45s
- Whereas on fine-tuning the parameters, we get the following output:
  ROC for XGBoost model
  76.18585945322294
  CPU times: user 2.83 s, sys: 507 ms, total: 3.34 s
  Wall time: 3.34 s

C. CatBoost

- On implementing the model with default parameters, we get the following output:
  Best model validation accuracy: 0.6627
  CPU times: user 11min 47s, sys: 14min 32s, total: 26min 19s
  Wall time: 14min 23s
- Whereas on fine-tuning the parameters, we get the following output: Best model validation accuracy: 0.6892
  CPU times: user 11min 17s, sys: 13min 57s, total: 25min 14s
  Wall time: 14min 23s

XIII. References

References

[1] Kaggle, "Machine Learning competition for developing AI classroom, organized by Riiid" IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[2] Riiid kaggle Competition [Online]. Available: https://www.kaggle.com/c/riiid-test-answer-prediction

[3] Ed-net challenge [Online]. Available: https://www.ednetchallenge.ai/

[4] ITS Dataset [Online]. Available: https://www.its.dot.gov/data/

[5] TOEIC exam [Online]. Available: https://www.ets.org/toeic

[6] RAPIDS API[Online]. Available:https://github.com/rapidsai

[7] Light Gradient Boosting Machine [Online]. Available: https://lightgbm.readthedocs.io/en/latest/

[8] XGBoost [Online]. Available:https://xgboost.readthedocs.io/en/latest/

[9] Catboost [Online]. Available:https://catboost.ai/

Acknowledgment

We would like to thank our Professor, Dr. Shucheng Yu, for helping us with the selection of this problem, for guiding us throughout the duration of this project, as well as for providing us with all the skills necessary for delivering real time Machine Learning based solutions for improving the education industry through this project.

We are also grateful to ednetchallenge.ai for introducing this environment for the sole purpose of making AI based personalized learning modules, more accessible, as well as for encouraging Machine Learning students and enthusiasts across the world.