MathE Dataset Analysis with XGBoost and Deep Learning Models

Yug Patel
Department of Computer Science
Kent State University
Kent, OH
ypatel10@kent.edu

Abhishek Pandit
Department of Computer Science
Kent State University
Kent, OH
apandit3@kent.edu

Anand Aggarwal
Department of Aeronautics and Engineering
Kent State University
Kent, OH
anand@kent.edu

Mustafa Alsenaidi
Department of Aeronautics and Engineering
Kent State University
Stow, OH
malsenal@kent.edu

***Abstract*— This paper provides the first student performance prediction model of MathE platform's math learning dataset using XGBoost Classifier. The paper presents models to predict if a student would answer a question correctly or incorrectly (Type of Answer variable) based on independent variables like Country, Topic, Subtopic, Question keywords and Question ID. We compute additional features like Question difficulty, Student skill, Country Skill and Topic difficulty. After hyper parameter tuning with learning rate and number of classifiers, the XGBoost classifier arrived at the best accuracy of *72.35%* with *500* classifiers and a learning rate of *0.0009*.**

Keywords*— *Math Learning datasets, MathE dataset, XGBoost Ensemble Model, Deep Learning

I. INTRODUCTION

Predicting student performance has several benefits both for educational institutions and students. Powerful predictive models can lead to early identification of students at higher risk of dropping out of the course or subject matter at hand. This can lead to timely interventions and improve overall educational experience. Consequently, such models can also aid in efficient resource allocation in terms of tutors, teaching assistants, supplemental materials and instructor's attention and efforts. Moreover, if a significant number of students are predicted to solve a particular problem or topic incorrectly, this can be a sign for the instructor to update their pedagogy for better outcomes. Hence with due consideration to data privacy, and by making sure the model is not biased in predicting a specific way for a particular group of students, such predictive models can be a handy tool for educational effectiveness.

II. LITERATURE REVIEW

The last few years have witnessed an increase in the use of machine learning (ML) and deep learning (DL) techniques to predict student performance. Arévalo-Cordovilla (2024) [1] employed Moodle LMS platform data to compare ML models for predicting success in online programming courses. Logistic Regression and Random Forest were among the top-performing models, with both high accuracy and interpretability. Similarly, Agyemang et al. (2024) [2] used school-level click-stream xAPI data and demonstrated that Random Forest provided well-balanced classification performance with an accuracy of 85.42% and a G-Mean of 0.9243. Such an approach is particularly useful due to its ease, the possibility of early intervention, and the ability to report actionable behavioral trends.

In the COVID-19 period, Junejo et al. (2024) [3] developed SAPPNet, a hybrid deep learning model that combined questionnaire data regarding digital tool usage, sleep patterns, and mental health. Their approach surpassed conventional ML and DL baselines by a significant margin with 93% accuracy. The significance of capturing static and temporal features in student data, especially during periods of academic disruption, is underscored by this study. However, the confines are in the computational demands of the model and its testing within one institution, causing problems of transferability.

Deep learning is used by Li and Liu (2021) [4] with a 1D CNN-LSTM model to extract complex time-series educational data. The model was supported using quantile transform and min-max scaling, achieving prediction accuracy of 59% MAE and 78% RMSE. Although such deep models hold much promise, they are harder to interpret and change without considerable tuning and domain knowledge.

[5] compares CNN and RNN-LSTM models of various structures with datasets across multiple universities indicated contradictory results. There was not one model suitable for all sets of data, especially when faced with class imbalance. While these deep learning practices enable early interventions for students more likely to fail, they often require more computational resources and sophisticated preprocessing than traditional ML models. This highlights ongoing demand for

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

balanced, scalable, and explainable models that are adaptable to individual school settings.

Only [6], [7] and [8] out of the analyzed previous literature, have worked with data from the MathE platform. Authors in [6] implement clustering algorithms to divide the questions on the MathE platform into 6 levels of difficulty beyond the "Basic" and "Advanced" level already present on the platform. Additionally, authors in [7] try to point out topics that need more attention on the platform, based on students' hit probability (probability of getting the question right). They apply the K-means clustering algorithm to assess country-wise student behavior. Paper [8] analyzes 99 limited students who attempted Linear Algebra questions in the Self Need Section (SNS) section of the platform. They observe that majority of those students attempt only basic difficulty questions and have a high error rate on both Basic and Advanced difficulty questions. These observations were made possible by using k-means clustering and nature inspired metaheuristic algorithms (Differential Evolution, Particle Swarm Optimization and Genetic Algorithm).

The authors would like to note that current literature lacks Boosting and Deep learning approaches to predict student performances in the MathE dataset. Here is where our work fits in the literature. Our XGBoost model provides capability of predicting student performance (Type of Answer variable) based on all other dependent variables like Student country, Question ID, Question Level, Math Topic, Subtopic and Keywords. This model is a step towards creating models to predict student performance for the benefits mentioned in Section I.

III. Dataset Description

The dataset was introduced in the paper [9] and accessed via the UC Irvine Machine Learning repository [11].

To analyze educational behavior, engagement levels, and academic outcomes, the dataset includes student learning activity logs and performance metrics. The core objective of the dataset is to cultivate virtual learning and foster knowledge exchange [10]. The MathE dataset was released in February 2024 and contains the mathematics questions attempted by students from February 2019 to December 2023. Each of the 9546 records represents a question attempted by a student.[10] Following are the distinct features belonging to each record as mentioned in [10].

Table I: Features in the Dataset

Feature Name	Feature Description
Student ID	A numerical ID given to each student at the time of registration on the platform.
Student Country	8 distinct countries from which the student is enrolled at the platform. The countries are Portugal, Lithuania, Italy, Ireland, Romania, Russia, Spain and Slovenia.
Question ID	A unique ID to identify 833 distinct questions.
Type of Answer	Correct (1) and Incorrect (0) * (see footnote)[1]
Question Level	Basic or Advanced, as decided by the volunteer Math professor.
Math Topic	15 distinct topics: Linear Algebra, Fundamental Mathematics, Graph Theory, Differentiation, Integration, Analytic Geometry, Complex Numbers, Differential Equations, Statistic, Real Functions of a Single Variable, Probability, Optimization, Set Theory, Numerical Methods
Math Subtopic	25 distinct subtopics belonging to each topic as follows: 1. Linear Algebra – Eigenvectors and Eigenvalues, Linear Systems, Linear Transformation, Matrices and Determinants, Vector Spaces 2. Fundamental Mathematics – Algebraic Expressions, Equations, and Inequalities; Elementary Geometry 3. Graph Theory – Only one subtopic named Graph Theory 4. Differentiation – Derivatives; Partial Differentiation 5. Integration – Integration Techniques; Definite Integrals; Double Integrals 6. Analytic Geometry – Only 1 subtopic named Analytic Geometry 7. Complex Numbers – Only 1 subtopic named Complex Numbers 8. Differential Equations – Differential Equations 9. Statistics – Statistics 10. Real Functions of a Single Variable – Domain, Image, and Graphics, Limits and Continuity 11. Probability – Probability 12. Optimization – Linear Optimization; Nonlinear Optimization 13. Set Theory – Set Theory 14. Numerical Methods – Numerical Methods
Question Keywords	194 distinct keywords, combination of which is used to represent each record.

[1](*) The dataset paper [1] has an error in defining whether they want to represent correct answers on the platform with 0 or 1. For our code implementation, we have assumed correct answers to be 1 and incorrect as 0.

There were no missing or repeating values found in the dataset.

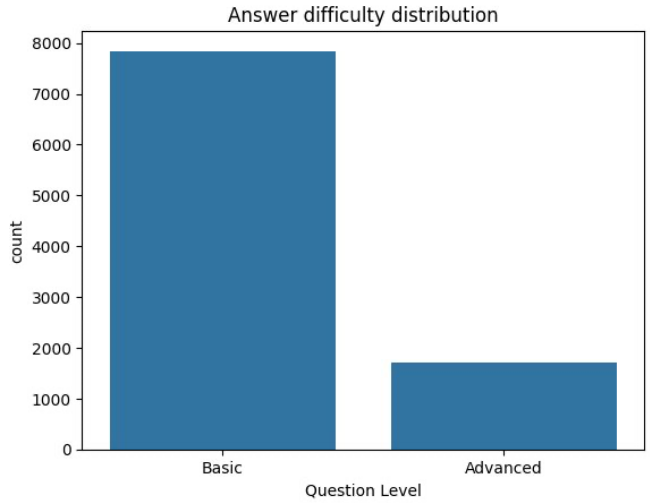

Fig 1: Total Count w.r.t Question Level

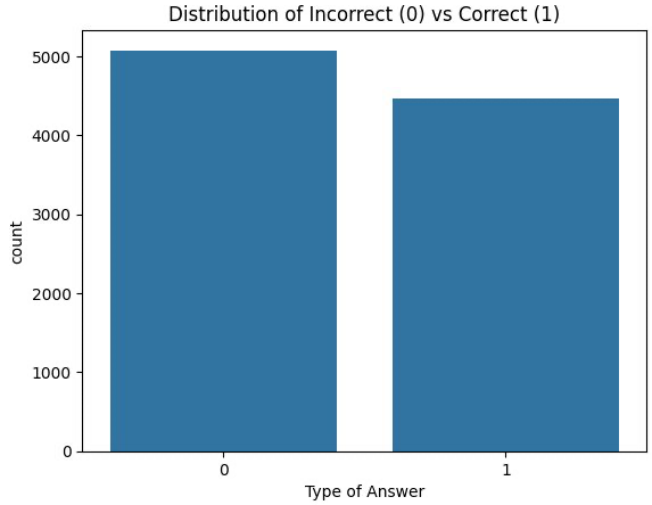

Fig 2: Total Count w.r.t Type of Answer

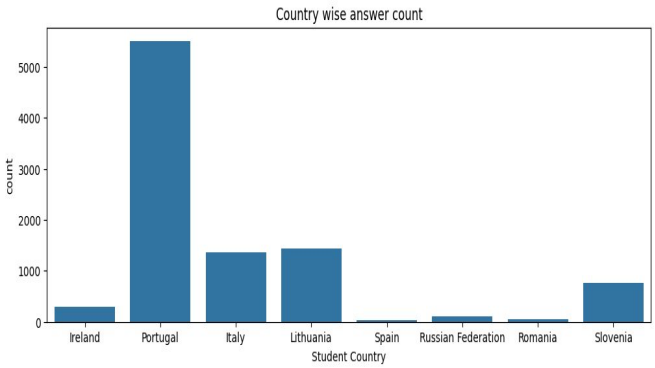

Fig 3: Total count w.r.t Student Country

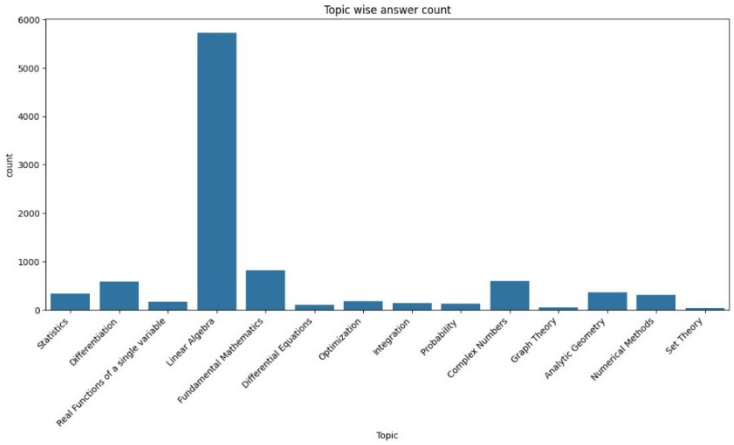

Fig 4: Total Count w.r.t Topic

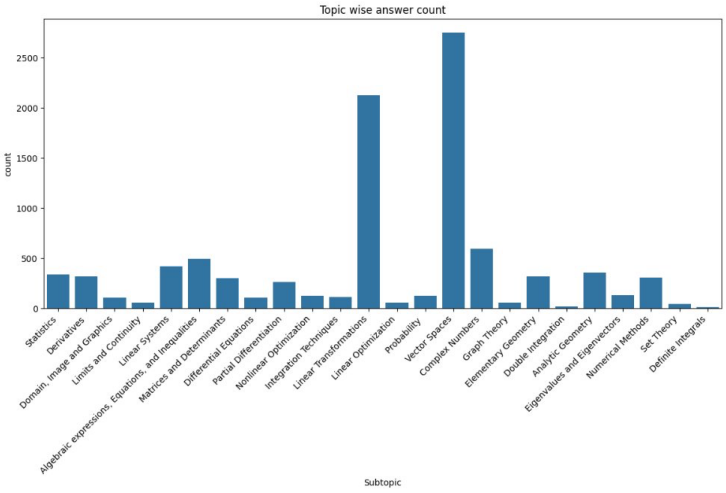

Fig 5: Total Count w.r.t Subtopic

IV. Data Preprocessing

To obtain more information per record, feature engineering was implemented by adding 4 new features as follows:

1. Student Skill (SS)
2. Question Difficulty (QD)
3. Country Skill (CS)
4. Topic Difficulty (TD)

Where the student skill has been calculated by taking the average correctness of each student across all answered questions.

$A_{i,j} \epsilon \{0,1\}$ correctness of student i i's answer to question j

S_i: set of all questions answered by student i.

$|S_i|$: total number of questions answered by student i.

$$StudentSkill_i = \frac{1}{|S_i|} \sum_{j \in S_i} A_{i,j}$$

This provides a rough marker for past performance of the student which can likely be a marker for future performance.

Question difficulty is defined as the average of the correctness rate per question as attempted by each student.

$A_{i,j}$: correctness of student *i*'s answer to question j

Q_i : set of all students who answered question j

$|Q_i|$: total number of answers submitted for question j

$$QuestionDifficulty_i = \frac{1}{|Q_i|} \sum_{j \in Q_i} A_{i,j}$$

Furthermore, Country Skill is the average performance of students in each country to get more information about the skills of each country separately.

$A_{i,j}$: correctness of student *i*'s answer to question j

C_k : set of all students where the student belongs to country k

$|C_k|$: total number of answers from students in country k

$$CountrySkill_k = \frac{1}{|C_k|} \sum_{(i,j) \in C_k} A_{i,j}$$

Since the dataset had different topics on which students were tested on, these topics could be relatively difficult or easier amongst themselves. This feature has been calculated by averaging the correctness of all answers given for questions under a specific topic.

$A_{i,j}$: correctness of student *i*'s answer to question j.

T_m : set of students associated with the topic m.

$|T_m|$: total number of answers given for questions in topic m.

$$TopicDifficulty_m = \frac{1}{|T_m|} \sum_{(i,j) \in T_M} A_{i,j}$$

These engineered features can provide good information that can help the model better understand the patterns of the different performance. By looking closely at the individual skills of the students, the difficulty of the questions, regional skills and the complexity of the specific mathematical theme can better enhance how the model can get more accurate predation across different students from different backgrounds and different math topics.

With the addition of the new features and the raw dataset a total of 12 features the model can utilize to get better accuracy. However, there are different types of features that the dataset has, which are split into three categories in preprocessing, Categorical Data (e.g. Country), numerical Data (e.g. Student skill) and Text Data which represent the keywords associated with each question. The distinguish between the different features is essential in the next step of the preprocessing.

After splitting the features to categorical, numerical and textual features, all of these features recombined using ColumnTransformer. This tool simultaneously applies different preprocessing methods to different feature types which then sum them up to a unified feature matrix [12]. However, since the categorical and the textual features are not numerical, they need to be transformed first. Categorical features such as student country and question level were converted using OneHotEncoder, which for each category creates a binary column which helps the model to interrupt them. As for the keywords they were processed using CountVectorizer / TfidfVectorizer. CountVectorizer transform the list of keywords into another matrix which represents the count of how many words are present in the dataset, where the TfidfVectorizer then take a step further by counting the how frequently specific words are present in the document [12]. Once the step of transformation of the categorical and the textual features is complete, they are then concatenated with the existing scaled numerical features by the ColumnTransformer which result in a comprehensive numerical representation of each data which is ready for the model to process. The following figure shows the pipeline of the feature engineering and preprocessing that results in the full matrix.

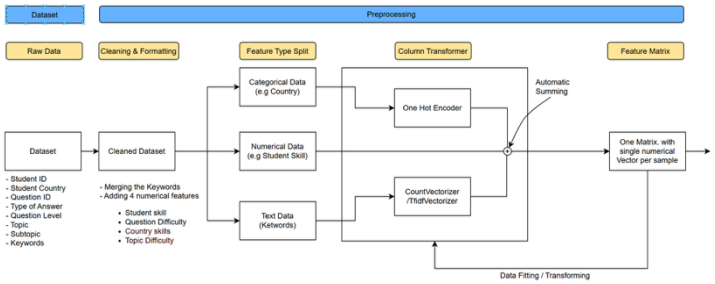

Fig 6: Feature engineering and Preprocessing Pipeline

V. Model Architecture

We leverage an eXtreme Gradient Boost (XGBoost) classifier [7] to obtain the highest accuracy. XGBoost is an implementation of the gradient-boost decision tree where XGBoost builds upon many small decision trees. Each tree then corrects the mistake of the previous tree where gradients are used to effectively learn, and all the trees combine (ensemble technique) to give better answers. Using different numbers of smaller trees (classifiers) and changing the learning rate we were able to get the most accuracy.

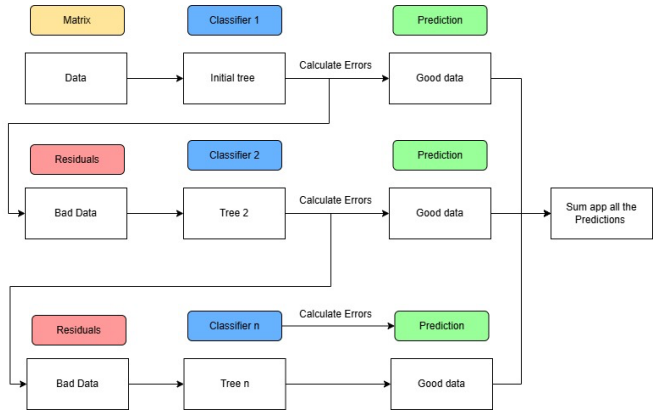

Fig 7: General overview of XGBoost architecture

VI. Results and Hyperparameter Tuning

The model's performance was evaluated by a learning curve and confusion matrix. The learning curve shows that the training accuracy started high at approximately ***82%*** and decreased gradually, topping out at around ***72.3%***

as more data was added (as evident in Figure 2). The testing accuracy (cross-validation score) started at approximately ***69%*** and rose gradually to around ***72%***. This is a sign that the model generalizes better as more training samples are added. The final test accuracy obtained by cross-validation since the average was ***0.7236***, which means that the model is still working in a stable and consistent manner on unseen data.

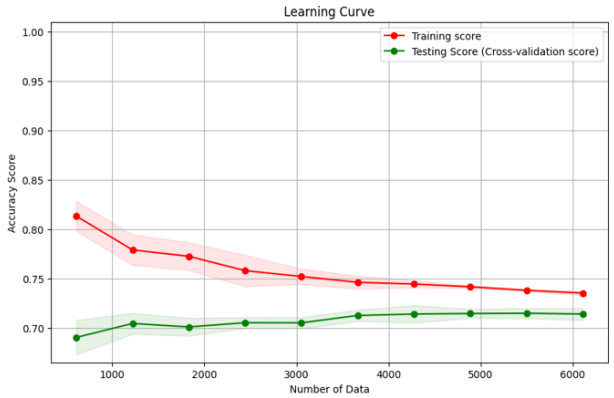

Fig 8: Training vs Testing Score

The confusion matrix also defines the classification ability of the model. In all the predictions, the model correctly identified ***791*** true negatives (label 0) and ***591*** true positives (label 1). While it produced ***225*** false positives (predicted 1 when it should have predicted 0) and ***303*** false negatives (predicted 0 when it should have predicted 1), the number of correct predictions is still far greater. However, the presence of false negatives and false positives indicates where it can be improved—primarily in reducing the misclassification of true positives.

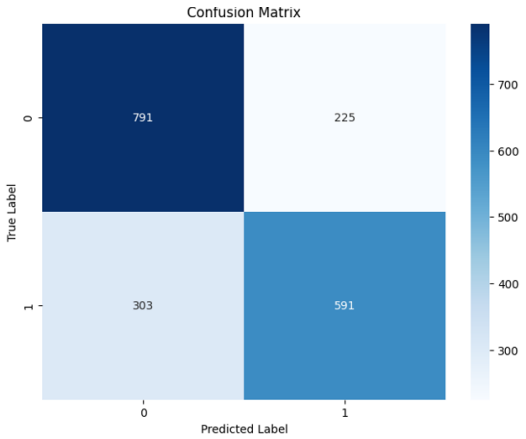

Fig 9: Confusion Matrix

VII. DISCUSSION

We consistently developed two different big models, one using TensorFlow Deep Learning sequential layers and the scikit-learn's XGBoost Classifier. The accuracy obtained using deep learning sequential model was around 59% and we generated 3 different XGBoost classifier models each having accuracy of around 68.32%, 69.32% and 72.35%. Since we were parallelly developing these models, in the end, we changed our primary focus to the XGBoost model which provided better results with lesser amount of computation than the deep learning model.

The authors would like to note that during the earlier iterations of training, our XGBoost model accuracy without engineering the 4 new features was lower than the ones mentioned above. Hence the need for adding those 4 features mentioned in Section IV and re-training.

In general, the results verify that the model behaves equally well with well-balanced generalization capacity, hence suitable for educational early prediction purposes. Nevertheless, the accuracy and misclassification rates at the moment suggest there is room for improvement through feature engineering, hyperparameter tuning, or the use of more advanced ensemble models.

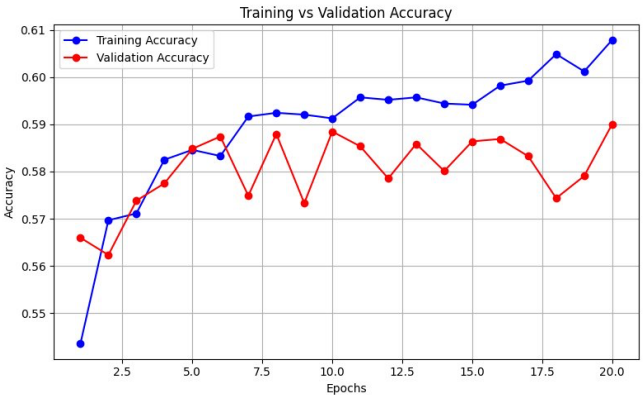

Fig 10: Deep Learning Sequential Model with accuracy 59%

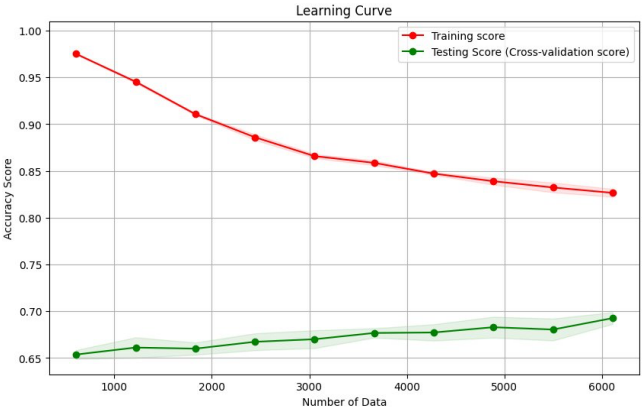

Fig 11: XGBoost with accuracy 68%.

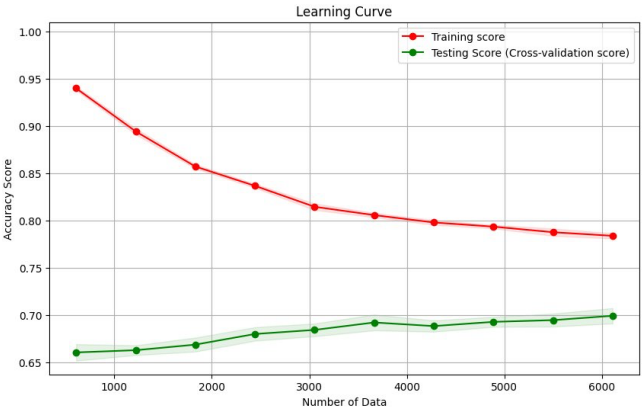

Fig 12: XGBoost with accuracy 69%

VIII. Conclusion

To conclude, this paper successfully develops and evaluates an XGBoost machine learning model to predict student performance on individual mathematics questions within the MathE dataset. Using XGBoost classifier model and countless hours of testing and fine tuning we were able to achieve an accuracy of 72.35% outperforming our Deep Learning sequential model with 59% accuracy. While this presents a new metric for the MathE platform, the results indicate scope for the future improvement better focused on feature refinement, exploring other machine learning techniques and tuning hyperparameters to further improve the accuracy.

References

[1] F. E. Arévalo-Cordovilla and M. Peña, "Comparative Analysis of Machine Learning Models for Predicting Student Success in Online Programming Courses: A Study Based on LMS Data and External Factors," Mathematics, vol. 12, no. 20, p. 3272, Oct. 2024, doi: https://doi.org/10.3390/math12203272.

[2] "Predicting Students' Academic Performance Via Machine Learning Algorithms: An Empirical Review and Practical Application," *Computer Engineering and Intelligent Systems*, Sep. 2024, doi: https://doi.org/10.7176/ceis/15-1-09.

[3] N. U. R. Junejo *et al.*, "SAPPNet: students' academic performance prediction during COVID-19 using neural network," *Scientific Reports*, vol. 14, no. 1, Oct. 2024, doi: https://doi.org/10.1038/s41598-024-75242-2.

[4] S. Li and T. Liu, "Performance Prediction for Higher Education Students Using Deep Learning," *Complexity*, vol. 2021, pp. 1–10, Jul. 2021, doi: https://doi.org/10.1155/2021/9958203.

[5] Abdallah Moubayed, MohammadNoor Injadat, Nouh Alhindawi, G. Samara, S. Abuasal, and Raed Alazaidah, "A Deep Learning Approach Towards Student Performance Prediction in Online Courses: Challenges Based on a Global Perspective," Dec. 2023, doi: https://doi.org/10.1109/acit58888.2023.10453917.

[6] B. F. Azevedo, Y. Amoura, A. M. A. C. Rocha, F. P. Fernandes, M. F. Pacheco, and A. I. Pereira, "Analyzing the MathE Platform Through Clustering Algorithms," *Lecture Notes in Computer Science*, pp. 201–218, 2022, doi: https://doi.org/10.1007/978-3-031-10562-3_15.

[7] [9] B. F. Azevedo, S. F. Romanenko, Maria, F. P. Fernandes, and A. I. Pereira, "Data Analysis Techniques Applied to the MathE Database," *Communications in Computer and Information Science*, pp. 623–639, Jan. 2023, doi: https://doi.org/10.1007/978-3-031-23236-7_43.

[8] Beatriz Flamia Azevedo, A. Maria, F. P. Fernandes, M. F. Pacheco, and A. I. Pereira, "Evaluating Student Behaviour on the MathE Platform - Clustering Algorithms Approaches," *Lecture notes in computer science*, pp. 319–333, Jan. 2022, doi: https://doi.org/10.1007/978-3-031-24866-5_24.

[9] Beatriz Flamia Azevedo, M. F. Pacheco, F. P. Fernandes, and A. I. Pereira, "Dataset of Mathematics Learning and Assessment of Higher Education Students Using the MathE Platform," *Data in brief*, pp. 110236–110236, Feb. 2024, doi: https://doi.org/10.1016/j.dib.2024.110236.

[10] T. Chen and C. Guestrin, "XGBoost: a Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, vol. 1, no. 1, pp. 785–794, Aug. 2016, doi: https://doi.org/10.1145/2939672.2939785.

[11] "UCI Machine Learning Repository," *Uci.edu*, 2024. https://archive.ics.uci.edu/dataset/1031/dataset+for+assessing+mathematics+learning+in+higher+education

[12] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Note:

Here is the github link which contains the jupyter notebook files and other assets for the project: https://github.com/Overfitters-Anonymous/MLDL-Final-Project.