NSCI 4051 – Workshop on Data Sciences

LAW Yiu Leung Eric (SID: 1155149315)

April 20, 2023

Topic: Predict Student Performance from Time Series Data

Instructor: Dr. Edmond Chan

Contents

List of Figures

List of Tables

Listings

1 Introduction

Game-based learning is a method of education that has seen growing popularity in recent years. It involves using gaming elements and mechanics to teach academic concepts and skills, making learning a more interactive and entertaining experience for students. This approach to education has been shown to be effective in engaging students and improving their academic outcomes.

The lack of knowledge tracing in game-based learning platforms is a missed opportunity to provide individualized support to students, which can ultimately improve their academic outcomes. Therefore, there is a need for increased focus on incorporating knowledge tracing techniques in educational games to support students' learning and development.

The project is trying to make advancement of knowledge-tracing methods for game-based learning. This will benefit the developers of educational games by providing them with valuable insights on how to create more effective learning experiences for their students. Ultimately, this work aims to enhance the quality of education through the use of game-based learning and promote better academic outcomes for students.

2 Dataset

Predict Student Performance from Game Play from Kaggle uses the Kaggle's time series API. Test data will be delivered in groupings that do not allow access to future data. The objective is to use time series data generated by an online educational game, Jo Wilder and the Capitol Case[1] (1), to determine whether players will answer questions correctly. There are three question checkpoints (level 4, level 12, and level 22), each with a number of questions. At each checkpoint, you will have access to all previous test data for that section. (2) There are 18 questions for each session, in `train_labels.csv`, it told whether the user for a particular session answered each question correctly.

`train.csv` contains 13,174,211 records with 20 features, the list of all features are included in appendix Training Dataset Features.

[1] A game based learning designed by PBS Wisconsin Education targeting child to learn English, history and arts.

Filename	Description	Rows	Columns
train.csv	training dataset	13,174,211	20
train_labels.csv	traning labels	212,022	2
test.csv	testing dataset	3,728	21
sample_submission.csv	sample of submission for prediction	54	3

Table 1: Datasets

Noted that `test.csv` and `sample_submission.csv` are only for submission in Kaggle competition[2], for the sake of simplicity of this project, those 2 datasets will be ignored in model development.

3 Exploratory Data Analysis

For complete and detailed exploratory data analysis, please refer to jupyter notebook `EDA.ipynb`, only important findings are included in this written report.

3.1 Missing Value

For *page*, *hover_duration*, *text* and *text_fqid*, they have high missing ratio over 60%. We may consider to drop these features later, but still we have to evaluate the relevance of them.

For *fullscreen*, *hq* and *music*, they are completely missing and contain no information, therefore, they have to be dropped.

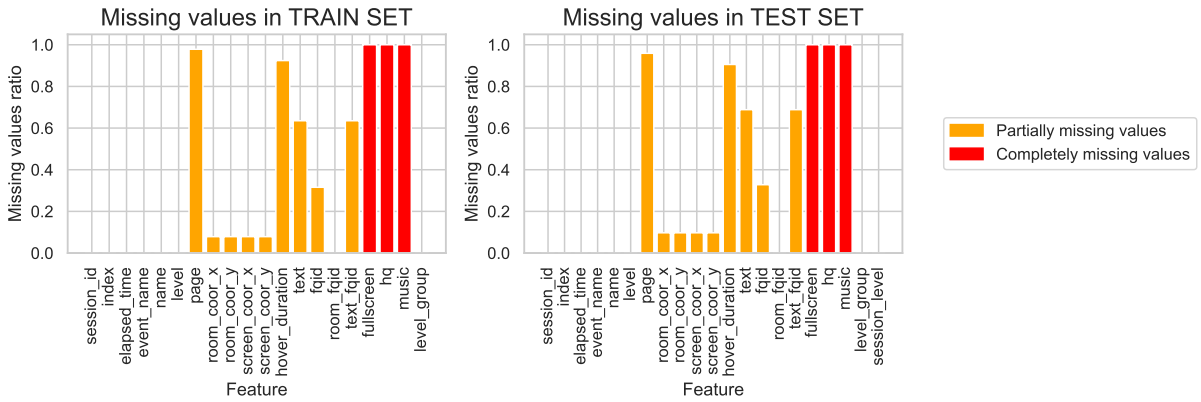

Figure 1: Missing Value

[2]Public competition that allows participants submit their prediction model.

3.2 Session and Index

There are 11,179 unique sessions in training dataset, each session have different number of events in range of [634, 19032], both the mean and median are at around 1,100, where the distribution appears to be normal with little positive skew. The 90th percentile is around 1,500, however, some session contain over 10,000 events, we may consider it as outlier.

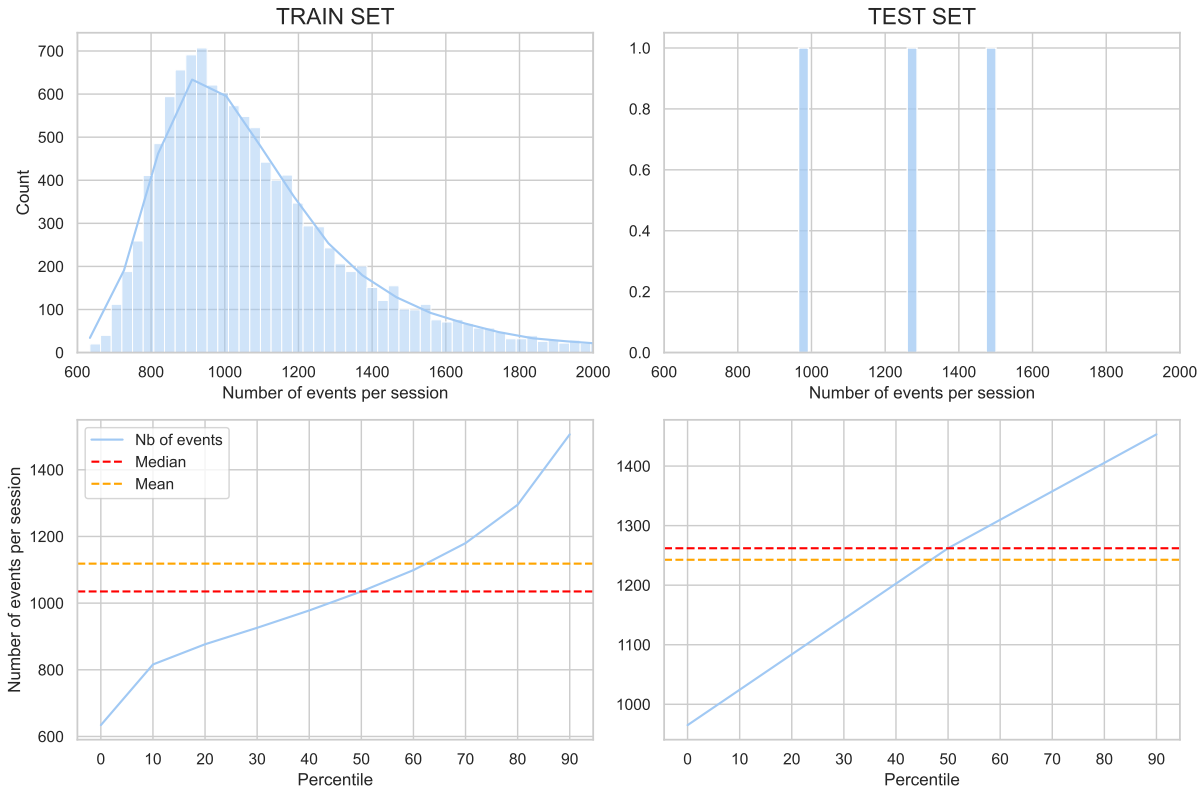

Figure 2: Session and Index

3.3 Elapsed Time

We may assume that elapsed time is positively correlated with the event index. As more time is spent, it is expected that a higher number of events will occur.

The figure show as expected, the elapsed time increases as the event index increases, which have strong correlation until the event index 1600 and no more correlations after 2800. This phenomenon can be explained by number of samples for larger event indexes are dropping significantly, those are likely to be outliers. Figure 4 shows 16 samples of outliers, most of the sessions have gaps between the increase of elapsed time, indicating pauses or inactivity. In the case of outlier sessions, the gap is huge due to extended periods of inactivity.

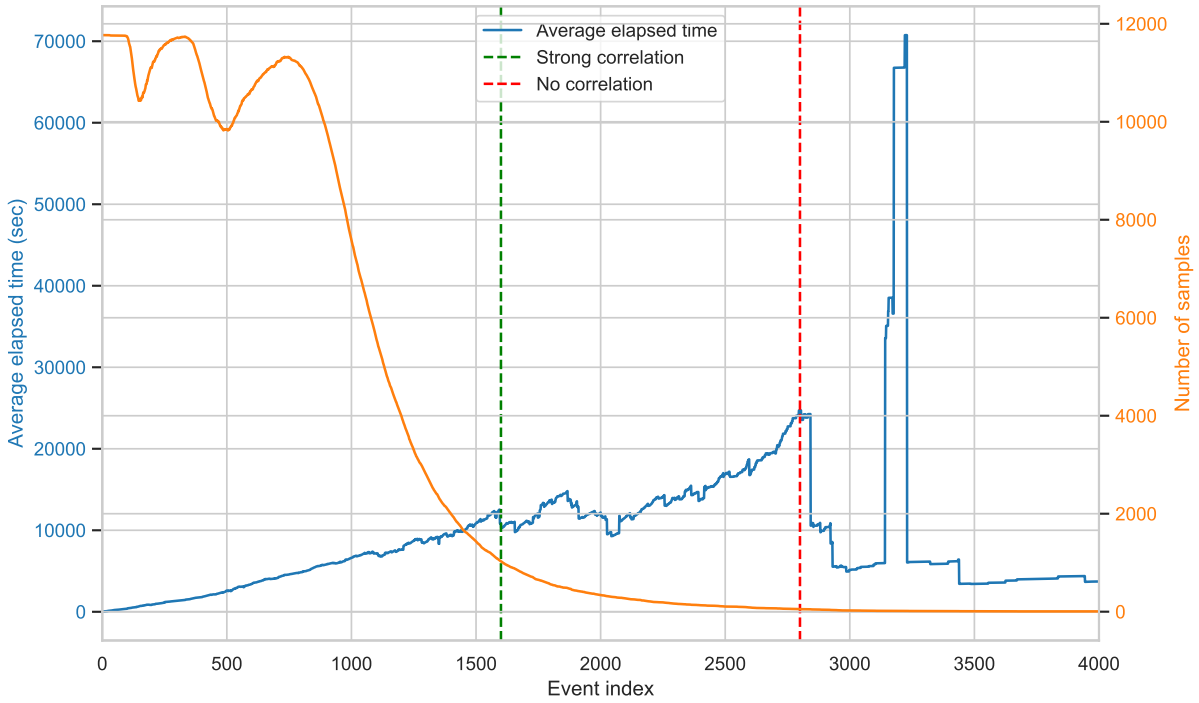

Figure 3: Elapsed Time

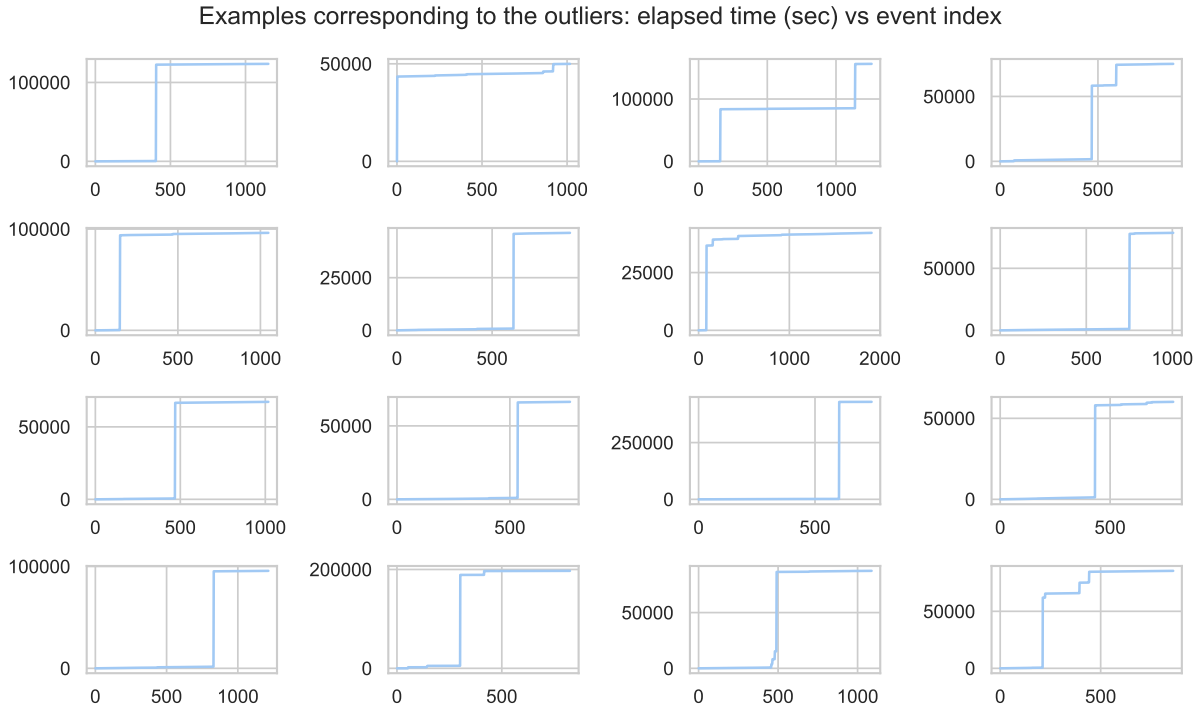

Figure 4: Elapsed Time of Outliers

3.4 Event Name and Name

There are 11 unique event names and 6 unique names, in the following bar charts and pivot table, we can see events are consist of 3 types, namely *checkpoint*, *click* and *hover*.

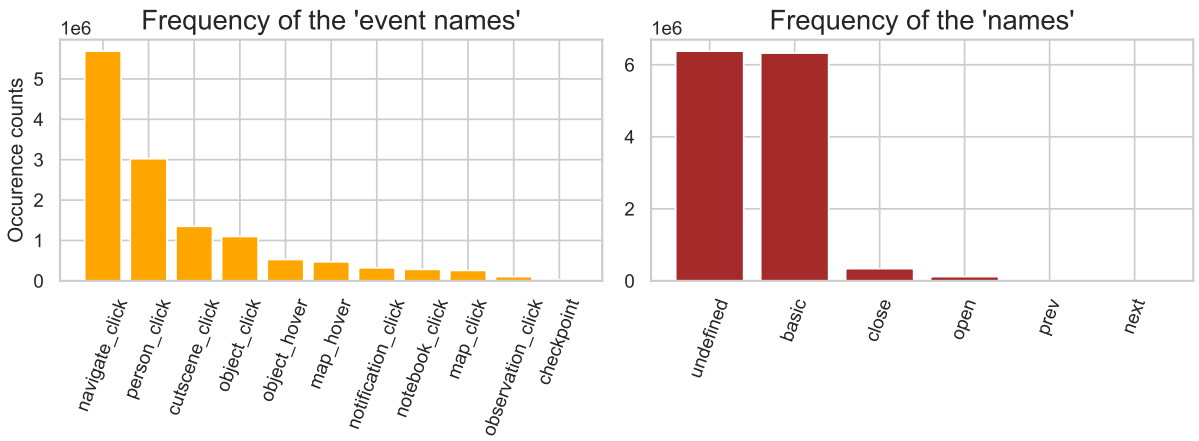

Figure 5: Event Names and Names

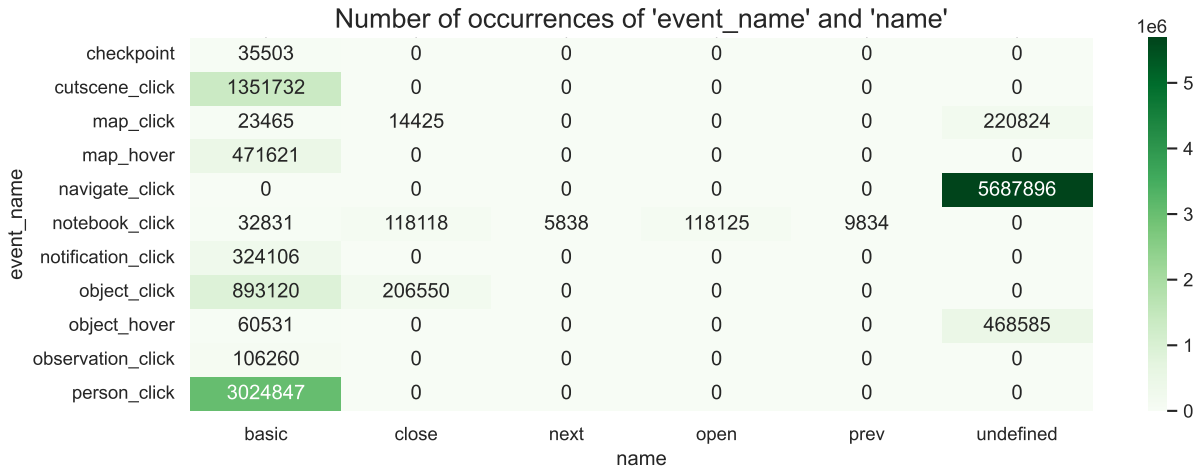

Figure 6: Event Names Pivot Table

3.5 Number of Log Data

Time series log data are recorded when an event is recorded, we could count the number of events by level to see the average number of events per question level.

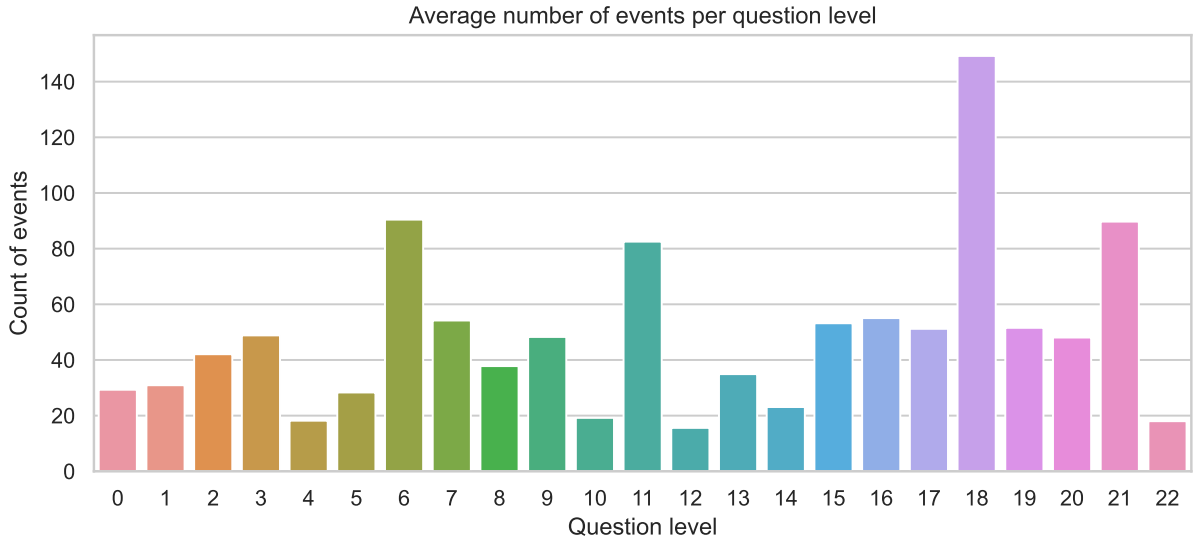

Figure 7: Average Number of Events per Question Level

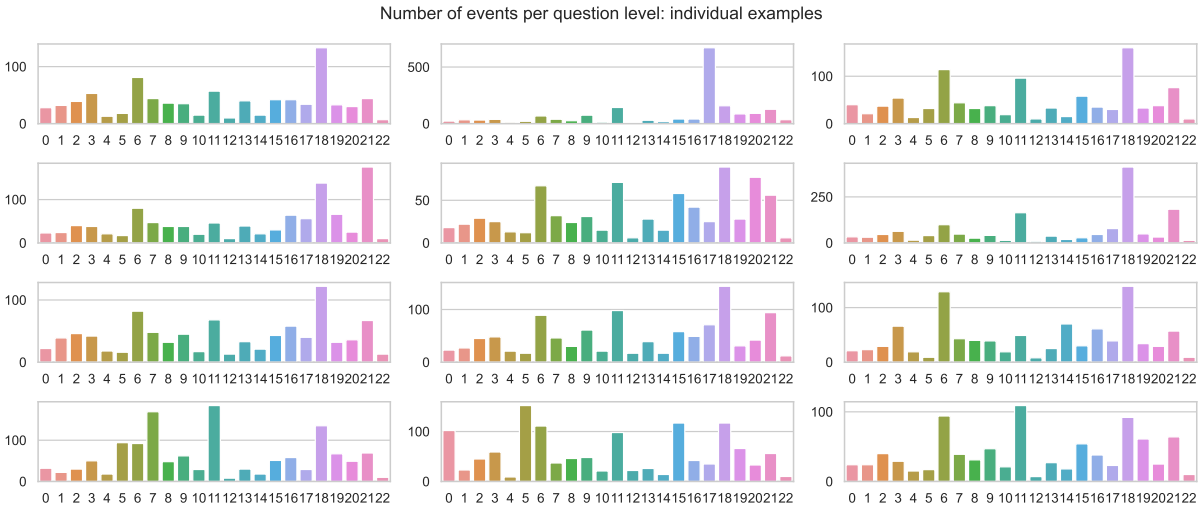

Figure 8: Average Number of Events per Question Level: Individual Examples

3.6 Coordinate Features

room_coor_x, *room_coor_y*, *screen_coor_x* and *screen_coor_y* are coordinate features. We can try to plot the coordinates and have a brief understanding of what they look like.

The patterns across different sessions are alike, where some noticeable clusters of clicks in certain regions. There are at least two potential button clusters, namely top left and bottom right. Players repeatedly navigated between these two clusters, resulting concentration of lines in the diagonal direction. These coordinates are probably a wealth of information, however in order to utilize it, that requires higher

machine learning techniques. For example, treat it as image and use computer vision to analysis.

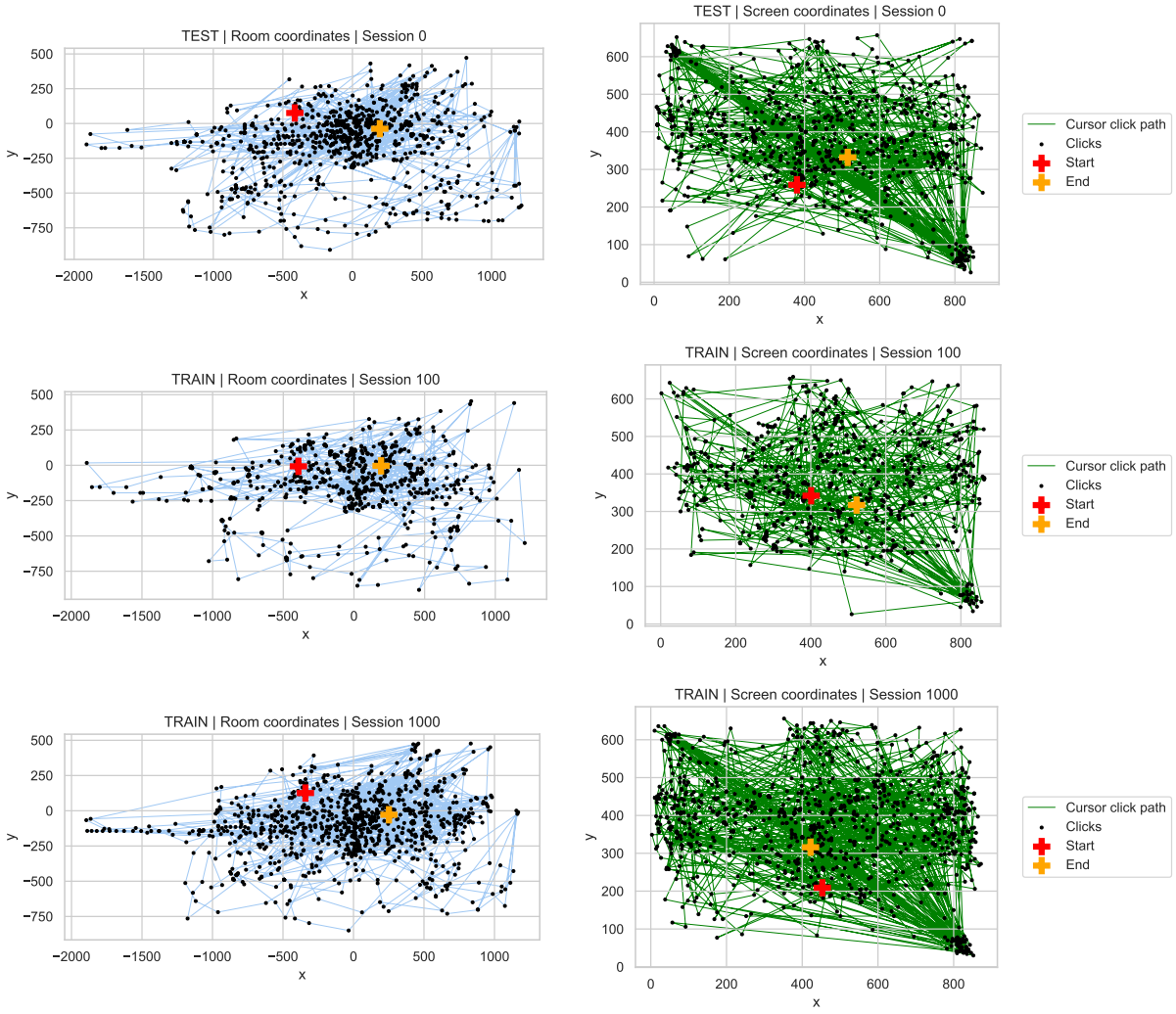

Figure 9: Coordinate Features

3.7 Labels

As the target of our prediction task, the goal is the predict the whether the session have correct answer on specific question. In total, about 70% correct and 30% incorrect, indicate that the dataset is slightly imbalanced.

Figure 10: Occurences of Labels

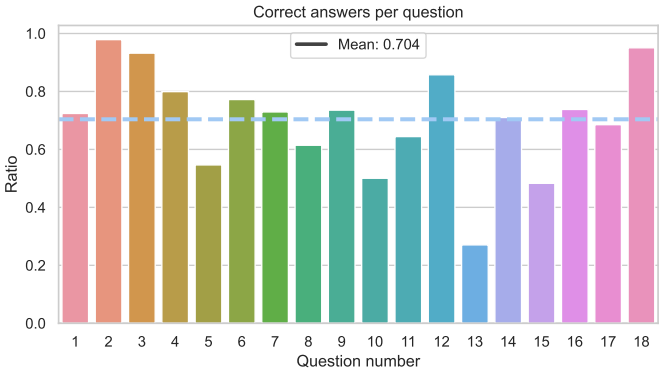

Figure 11: Correct Ratios

4 Data Preparation for Model Training

In order to use time series log data for model training, it is necessary to perform data preparation and feature engineering. This could be done by aggregating the data into a session-wise level, that is aggregating each session into one row, which can be used as input for the model. To handle this huge training dataset, Python library Polars[3] (3) is used instead of the commonly used **Pandas**, as **Polars** performs much faster on data manipulation on large dataset. Some processes are explained in the following subsection, for the whole procedure, please refer to `XGB_model.ipynb`.

4.1 Feature Engineering

4.1.1 Numerical Features

7 numerical features are selected, ['page', 'room_coor_x', 'room_coor_y', 'screen_coor_x', 'screen_coor_y', 'hover_duration', 'elapsed_time_diff']. Each numerical features are then used to compute additional statistical features such as various quantiles, minimum, maximum, mean, and standard deviation for each numerical feature. For elapsed time and coordinates, new columns are calculated base on the different from last record over ["session_id", "level_group"]

4.1.2 Categorical Features

5 categorical features are selected, ['event_name', 'name', 'fqid', 'room_fqid', 'text_fqid']. Minimum, maximum, mean, and standard deviation are then calculated for the elapsed time difference over each categorical

[3]Polars is a lightning fast DataFrame library/in-memory query engine. Its embarrassingly parallel execution, cache efficient algorithms and expressive API makes it perfect for efficient data wrangling, data pipelines, snappy APIs and so much more.

features. Furthermore, some further features are calculated based on specific group level.

4.2 Preprocessed Datasets

The feature engineering procedure is separately perform by *level_group*: ['0-4', '5-12', '13-22'] which is the level group of the question, then 3 datasets are generated. After dropping features with $> 90\%$ missing ratio and features only have single unique value, the datasets contain 599, 914 and 1059 features respectively in each level group, all contain 11,779 rows representing every unique sessions.

$$\text{datasets } S^{(n)} = \left(x_i^{(n)}, y_i^{(n)}\right) \text{ for } i = 1, ..., N^{(n)}$$
$$x_i^{(n)} = \left(x_i^{(n,1)}, x_i^{(n,2)}, ..., x_i^{(n,K_n)}\right) \text{ for } K \text{ features in } n^{\text{th}} \text{ dataset}$$
$$\text{where } y_i^{(n)} \text{ is the target label}$$

5 Machine Learning Model Approach

The challenge at hand can be framed as a binary classification problem, wherein the goal is to accurately classify player in each question into one of two possible categories, namely correct and incorrect. Various machine learning models can be utilized to address this task, including logistic regression, deep neural networks, random forest, etc. Given the need to strike a balance between model complexity and computational efficiency, we have chosen to utilize a random forest model in this project. This approach will enable us to effectively and efficiently classify instances based on relevant features and their associated information gain.

For this classification problem, we have 18 questions to classify, in subsection 3.5 and 3.7 shows that there are some patterns regarding question level. It is safe to assume that different questions have various difficulties and features have different level of importance to each question, therefore, it is decided to split the model into 18 individual models.

5.1 Random Forest Model Specifications

Random forest is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of the overall model (4). In a random forest, a set of decision trees are trained on different subsets of the training data, and each tree is allowed to make its own prediction. The final

prediction is then determined by combining the predictions of all the trees in the forest. This approach helps to reduce the impact of overfitting and noisy data, and improves the generalization performance of the model. Figure 12 illustrate the concept (5).

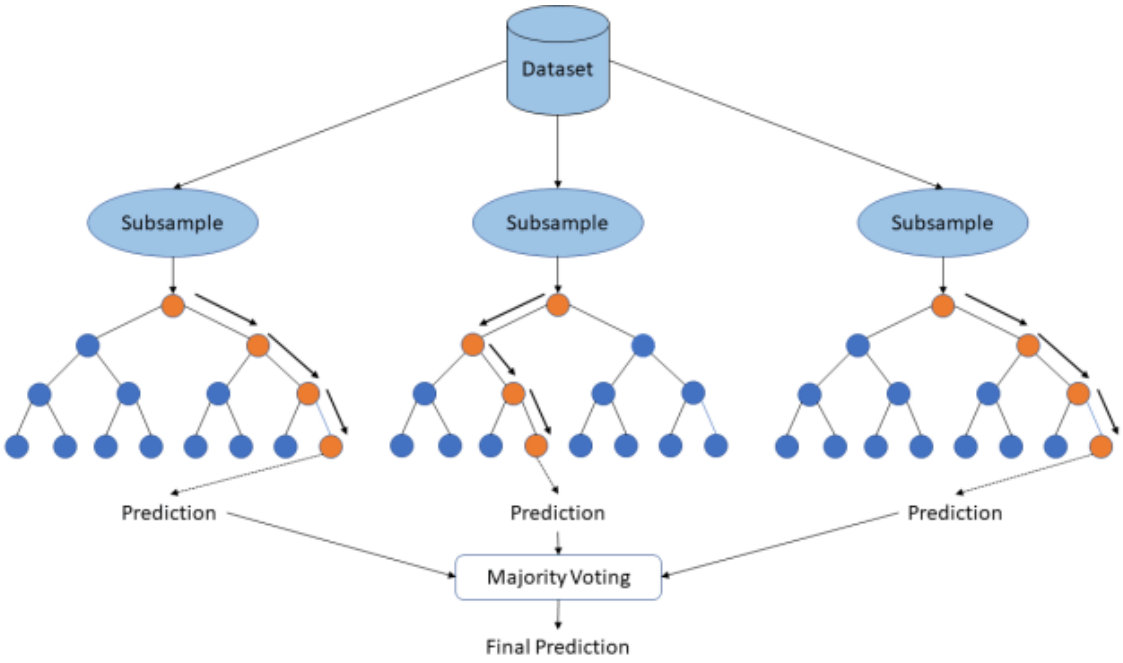

Figure 12: Random Forest

To implement random forest in Python, `XGBoost` (6) library is used, and the parameters are as following:

```
xgb_params = {
    'objective' : 'binary:logistic',
    'eval_metric':'logloss',
    'learning_rate': 0.05,
    'max_depth': 4,
    'n_estimators': 1000,
    'early_stopping_rounds': 50,
    'tree_method':'hist',
    'subsample':0.8,
    'colsample_bytree': 0.4,
    'use_label_encoder' : False
}
```

Listing 1: XGBoost Parameters

Noted that *early_stopping_rounds* is set at 50, that stops the training of the model when the performance

on a validation set stops improving. This can help to prevent overfitting by stopping the training before the model starts to fit the training data too closely.

5.2 Group Fold Cross Validation

Overfitting is a common problem in machine learning, where a model becomes too complex and starts to fit the training data too closely. This means that the model is not able to generalize well to new, unseen data, and its performance may suffer. Overfitting can occur when a model is too complex for the amount of data available, or when the model is too flexible and captures noise or irrelevant features in the data. To prevent overfitting, we can use cross-validation (CV) by evaluating the performance of a model on new, unseen data. Group fold cross-validation is a variant of k-fold cross-validation. In group fold cross-validation, the data is divided into K folds as in Figure 13 (7), but the folds are created in such a way that each fold contains one or more groups, and the groups are not split across the folds. This ensures that the model is tested on data that is truly independent of the data it was trained on.

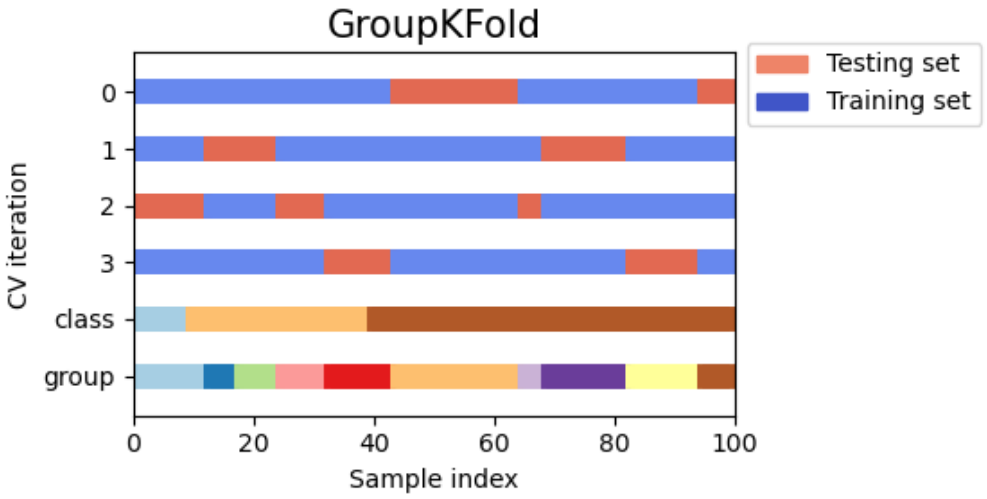

Figure 13: Group K Fold Cross Validation

Let set $K = 5$ and adopt 5 groups fold cross-validation, in this case, $5 \times 18 = 90$ model training are required to find the optimal models.

6 Model Evaluation

Once the model training is completed, it is necessary to perform model evaluation and compute the cross-validation F1 scores. This step is crucial in assessing the performance of the trained model and determining its ability to generalize to new data.

6.1 Optimal Prediction Threshold

Since the random forest is set on binary logistic classification, that is going to return a probability p on whether the target is 0 or 1 based on a threshold t, by default, the threshold $t = 0.5$. However, the threshold t can be set by ourself, such that we can find optimal threshold such that F1 score is optimized. The range of threshold is set be to [0.4, 0.8] as a reasonable range.

$$0 \leq p_i \leq 1$$

$$0.4 \leq t_i \leq 0.8$$

$$\text{target } \hat{y_i} = \begin{cases} 1 & \text{if } p_i \geq t_i \\ 0 & \text{if } p_i < t_i \end{cases}$$

6.2 F1 Score

confusion matrix can show the summary of prediction and will be used to calculate accuracy, precision, recall and F1-score.

	True Class	
Predicted Class	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

Table 2: Confusion Matrix

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{and} \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

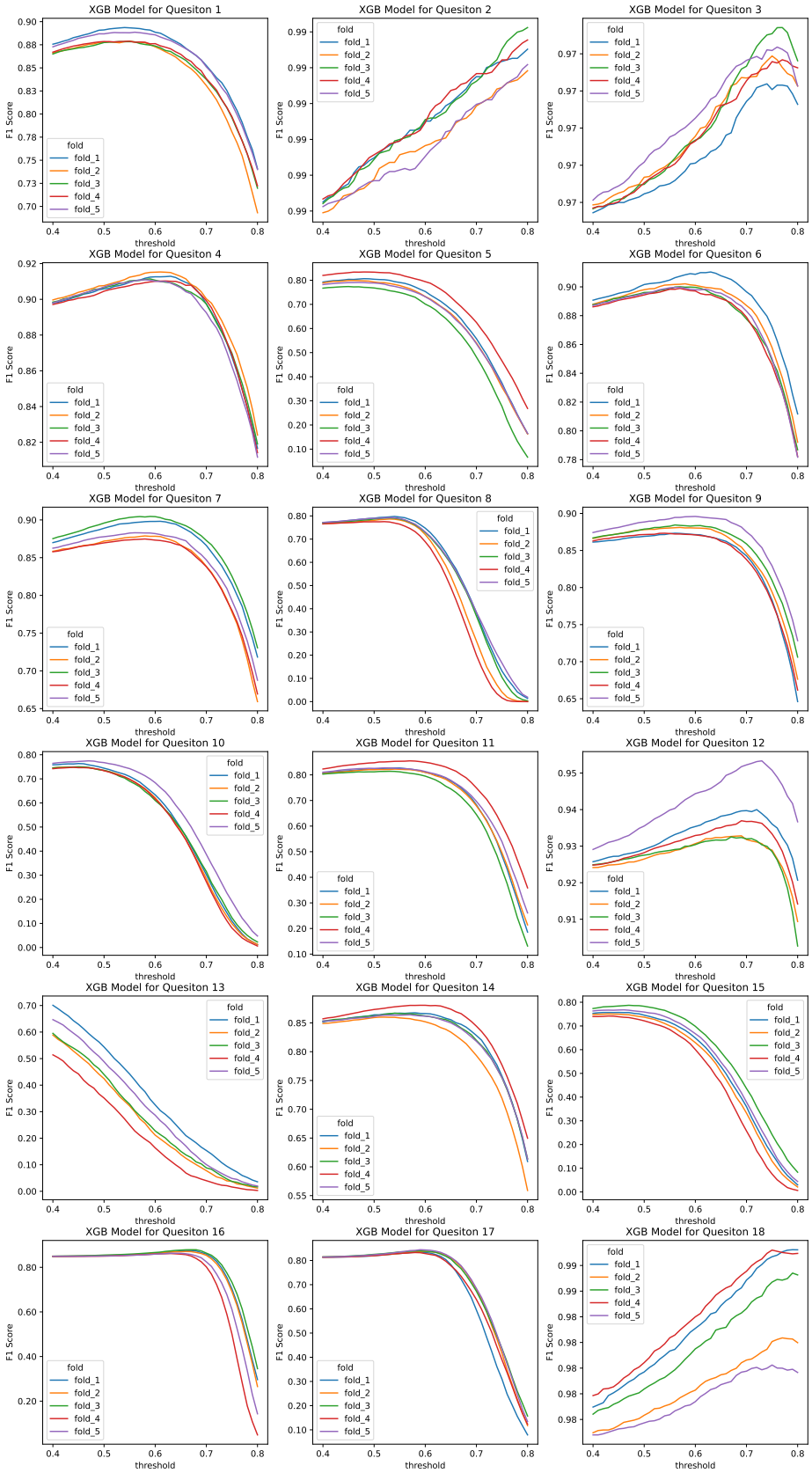

Figure 14: F1 Score vs Threshold

The best F1 Scores of each questions and folds:

Question	fold 1	fold 2	fold 3	fold 4	fold 5	Optimal Threshold
1	**0.8936**	0.8788	0.8785	0.8789	0.8886	0.54
2	0.9945	0.9939	**0.9951**	0.9948	0.9941	0.80
3	0.9724	0.9739	**0.9754**	0.9737	0.9744	0.77
4	0.9129	**0.9153**	0.9114	0.9103	0.9109	0.61
5	0.8065	0.7989	0.7738	**0.8342**	0.7908	0.48
6	**0.9103**	0.9021	0.9001	0.8989	0.8996	0.63
7	0.8982	0.8788	**0.9047**	0.8748	0.8830	0.59
8	**0.7978**	0.7865	0.7913	0.7753	0.7923	0.54
9	0.8733	0.8811	0.8844	0.8733	**0.8960**	0.60
10	0.7635	0.7485	0.7501	0.7478	**0.7747**	0.47
11	0.8270	0.8225	0.8140	**0.8548**	0.8274	0.57
12	0.9400	0.9328	0.9325	0.9369	**0.9533**	0.73
13	**0.7009**	0.5888	0.5952	0.5141	0.6468	0.40
14	0.8676	0.8600	0.8669	**0.8806**	0.8641	0.59
15	0.7568	0.7506	**0.7871**	0.7419	0.7678	0.47
16	0.8751	0.8708	**0.8789**	0.8619	0.8631	0.67
17	0.8346	0.8374	0.8395	0.8328	**0.8445**	0.59
18	**0.9892**	0.9824	0.9874	0.9892	0.9802	0.79
Overall			0.8930			

Table 3: F1 Scores

Most of the questions have convex optimal point for threshold, except question 2 and 13, their F1 score just keep increasing and decreasing respectively. That could be explained by Figure 11, as question 2 have highest correct ratio and question 13 have lowest correct ratio. Due to the distribution of significantly imbalanced datasets, the models are biased to predict them all correct or all incorrect. Generally speaking, The correct ratio and optimal threshold are positively correlated, higher correct ratio makes the optimal threshold higher and vice versa.

7 Conclusion

In conclusion, the results of this study demonstrate that using random forest models for binary classification tasks can be effective when considering sub-groups. The models trained for each questions showed promising results with F1 scores ranging from 0.7009 to 9951, in whole, all models combined achieve an overall F1 score 0.8930. These findings suggest that the model has the potential to be used in practical applications for each sub-group. However, it is important to continue monitoring the model's performance over time to ensure that it remains effective and accurate.

In addition to training and evaluating the random forest models, feature engineering was also conducted to improve the performance of the models. The feature engineering process involved the creation of new features based on data exploration techniques. These new features were then incorporated into the models to improve their predictive power.

7.1 Limitations and Potential Improvement

The main limitation of this study is the presence of imbalanced dataset. Specifically, there were significantly more instances of one class than the other, which can lead to biased model performance and inaccurate predictions. That could be further improved by advanced techniques on imbalanced training and finding the best classification threshold in imbalanced classification (8).

Another potential improvement is to convolutional neural networks (CNNs) for time series analysis (9). Using CNNs for time series analysis is a promising area for future research and may lead to improved predictive performance in this domain. It is important to note, however, that the implementation of CNNs can be computationally expensive and may require more training data than other models. Therefore, careful consideration and experimentation would be needed to determine the optimal approach for this task.

8 Bibliography

References

[1] P. W. Education, "Jo wilder and the capitol case." https://pbswisconsineducation.org/jowilder/about/, 2021. Accessed: 2023-03-18.

[2] T. L. A. Lab, "Predict student performance from game play." https://www.kaggle.com/competitions/predict-student-performance-from-game-play/overview, 2023. Accessed: 2023-03-18.

[3] R. Vink, "Polars: Blazingly fast dataframes." https://www.pola.rs/, 2017. Accessed: 2023-03-25.

[4] A. Cutler, D. R. Cutler, and J. R. Stevens, *Random Forests*, pp. 157–175. New York, NY: Springer New York, 2012. 10.1007/978-1-4419-9326-7_5.

[5] B. Padovese and L. Padovese, "A machine learning approach to the recognition of brazilian atlantic forest parrot species," p. 5, 12 2019. 10.1101/2019.12.24.888180.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016. 10.1145/2939672.2939785.

[7] scikit-learn developers, "Visualizing cross-validation behavior in scikit-learn." https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html#sphx-glr-auto-examples-model-selection-plot-cv-indices-py, 2022. Accessed: 2023-03-27.

[8] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, "Finding the best classification threshold in imbalanced classification," *Big Data Research*, vol. 5, pp. 2–8, 2016. 10.1016/j.bdr.2015.12.001.

[9] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017. 10.21629/JSEE.2017.01.18.

[10] MACHENGYUAN, "Simple xgb model." https://www.kaggle.com/code/machengyuan/simple-xgb-model/notebook, 2023. Accessed: 2023-03-25.

[11] P. Bacher, "Predict student performance from game play." https://www.kaggle.com/code/paulbacher/detailed-eda-student-perf-from-game-play, 2023. Accessed: 2023-03-23.

9 Appendix

9.1 Training Dataset Features

Feature	Description
session_id	the ID of the session the event took place in
index	the index of the event for the session
elapsed_time	time has passed (in ms) between the start of the session and when the event was recorded
event_name	the name of the event type
name	the event name (e.g. identifies whether a notebook_click is is opening or closing the notebook)
level	what level of the game the event occurred in (0 to 22)
page	the page number of the event (only for notebook and related events)
room_coor_x	the coordinates of the click in reference to the in-game room (only for click events)
room_coor_y	the coordinates of the click in reference to the in-game room (only for click events)
screen_coor_x	the coordinates of the click in reference to the player's screen (only for click events)
screen_coor_y	the coordinates of the click in reference to the player's screen (only for click events)
hover_duration	how long (in milliseconds) the hover happened for (only for hover events)
text	the text the player sees during this event
fqid	the fully qualified ID of the event
room_fqid	the fully qualified ID of the room the event took place in
text_fqid	the fully qualified ID of the
fullscreen	whether the player is in fullscreen mode
hq	whether the game is in high-quality
music	whether the game music is on or off
level_group	which group of levels - and group of questions - this row belongs to (0-4, 5-12, 13-22)

Table 4: All Features of Training Dataset