

Kaggle competition: EDM Cup 2023

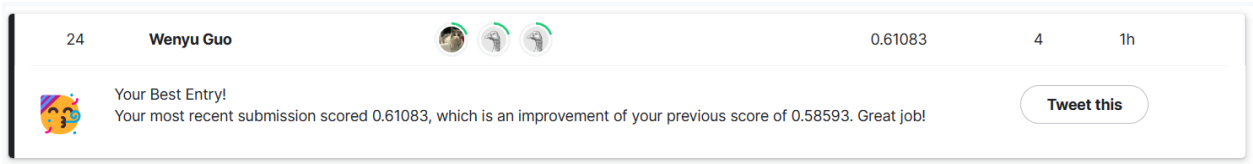
Predict students' performance on end-of-unit math problems using their clickstream data from previous assignments

Xinxue Lu, Wenyu Guo, Jing Du

Link to the submission on Kaggle

<https://www.kaggle.com/competitions/edm-cup-2023/leaderboard>

Team: Wenyu Guo

A screenshot of a Kaggle submission card. At the top, it shows the rank '24', the name 'Wenyu Guo', three profile icons, the score '0.61083', and the number of submissions '4' with a '1h' time indicator. Below this, a message says 'Your Best Entry! Your most recent submission scored 0.61083, which is an improvement of your previous score of 0.58593. Great job!'. There is a 'Tweet this' button on the right.

Definition of the prediction task

Base on the Kaggle website link, it mentioned that: The prediction task is to generate a binary measure of problem correctness using the given data. The evaluation metric for this competition is the Area Under the Receiver Operating Characteristic Curve (AUC). A higher AUC indicates better performance in predicting the binary measure of problem correctness.

To achieve this, we will train different models using the cleaned data and compare their performance using the AUC evaluation metric.

Datasets used in the final submission

In this prediction task, we primarily select variables from the action_logs.csv and problem_detail.csv to predict students' performance. The chosen variables include timestamp for the instant the action was taken on a problem, the maximum number of attempts, the measures students take on a problem (such as request hints, request tutoring), the problem type, and whether the problem includes images, formulas, or videos.

#	Column	Data type
0	timestamp	float64
1	max_attempts	float64
2	action	object
3	problem_type	object
4	problem_contains_image	float64
5	problem_contains_equation	float64
6	problem_contains_video	float64

7	action_count	float64
---	--------------	---------

1. **Timestamp:** It can provide valuable information about students' learning behaviors and preferences. By capturing time-based patterns, the 'timestamp' variable can help improve the predictive power of our model and provide a more accurate representation of students' performance on individual problems.
2. **Maximum Attempts:** A higher number of attempts to solve a problem might signal that a student faces challenges in understanding the problem, which could affect their chances of solving it correctly.
3. **Actions:** Students who seek hints or other assistance might have a higher likelihood of solving the problem correctly as they receive additional support. This variable can help predict problem correctness by evaluating students' problem-solving approaches and the impact of external assistance. The "action" variable is an important feature that captures the various actions students take while completing their assignments. To enhance the information contained in this variable, we will also include the "score_viewable" and "continuous_score_viewable" variables, as their values are only present for the "problem_started" action.
4. **Problem Type:** Different problem types might have varying levels of difficulty and cognitive demands, which can impact a student's likelihood of solving them correctly. Including problem type in the prediction can help tailor the model to account for these variations.
5. **Problem Content (Images, Formulas, Videos):** The presence of images, formulas, or videos might affect a student's comprehension and engagement, which could impact their ability to solve the problem correctly. By considering different content presentations, the prediction model can account for their influence on problem correctness.
6. **Action_count:** We have introduced a new variable, Action_count, into our model to predict student performance on individual problems more accurately. Action_count represents the quantity and frequency of actions taken by a student in the past and is derived from the example code provided on the Kaggle website. It captures student engagement, reflects learning strategies, monitors progress over time, and enhances the model's predictive power.

And then, a critical aspect of the preprocessing phase is converting categorical variables into numerical representations. We will use the "get_dummies" function for handling categorical variables in the datasets used for the final submission:

1. **Identify Categorical Variables:** Before applying the "get_dummies" method, we must first identify the categorical variables in the dataset that require conversion. We use "df.info()" to figure out which variables should be convert. These variables include action, problem_multipart_id, problem_type, problem_skill_description.
2. **Convert Categorical Variables:** Use the "get_dummies" function from the pandas library to convert the identified categorical variables into dummy variables. This method creates binary columns for each category of the original variable, with a value of 1 indicating the presence of that category and 0 otherwise. This approach is also known as one-hot encoding.

Data preprocessing and feature engineering

The data preprocessing and feature engineering stage is crucial for developing accurate and reliable models. As we mentioned before, we primarily select variables from action_logs.csv and problem_detail.csv to predict students' performance based on the problem_id variable. This section will discuss the process of handling irrelevant and redundant variables and managing missing data.

#	Columns	Data type
0	assignment_log_id	object
1	timestamp	float64
2	problem_id	object
3	max_attempts	float64
4	available_core_tutoring	object
5	score_viewable	float64
6	continuous_score_viewable	float64
7	action	object

8	hint_id	object
9	explanation_id	object
10	problem_multipart_id	object
11	problem_multipart_position	float64
12	problem_type	object
13	problem_skill_code	object
14	problem_skill_description	object
15	problem_contains_image	float64
16	problem_contains_equation	float64
17	problem_contains_video	float64
18	problem_text_bert_pca	object

Handling Variables

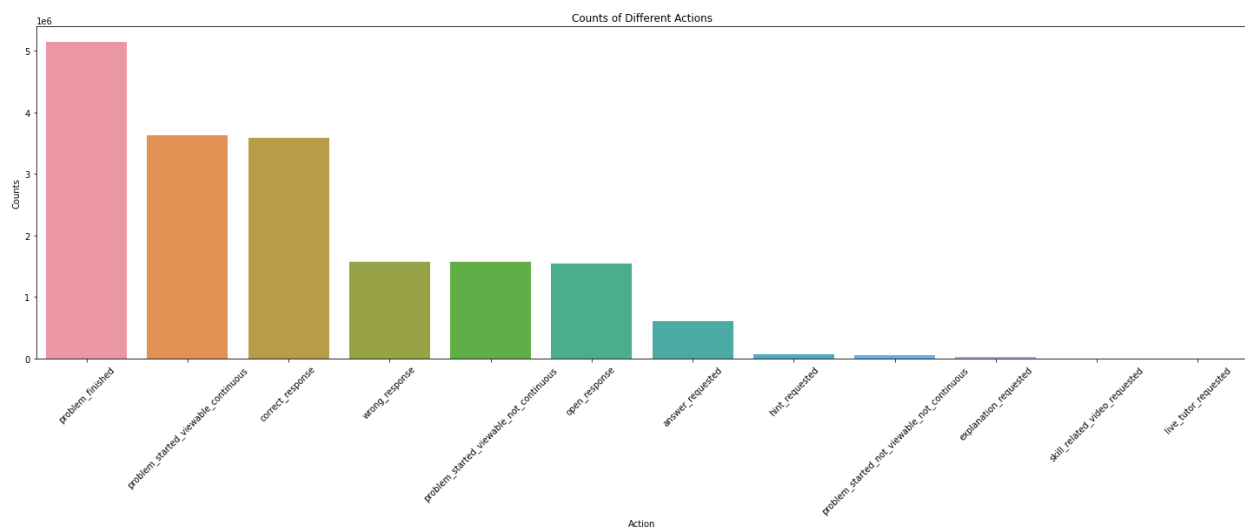
1. Removing Irrelevant Variables: We identified two irrelevant variables which are 'hint_id' and 'explanation_id,' because we do not use hint_detail.csv and explanation_detail.csv in our analysis. Removing these variables simplifies the dataset and reduces noise.
2. Removing Redundant Variables: The 'problem_skill_code' is not useful because its information is already captured in the 'problem_skill_description'. By removing redundant variables, we can reduce multicollinearity and improve the interpretability of our models.
3. The 'problem_multipart_id' and 'problem_multipart_position' variables provide information about the multi-part structure of problems. However, this information might not be directly related to a student's performance on individual problems, limiting their predictive power for the target variable. Moreover, Including too many variables in the model can lead to increased computational requirements, making it challenging to run the model on machines with limited resources. Removing these variables can help reduce the computational burden, ensuring that the model can be successfully run on our current hardware setup.

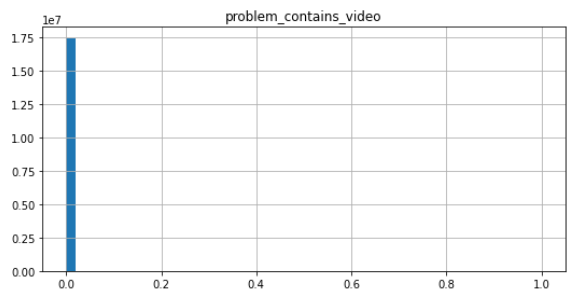
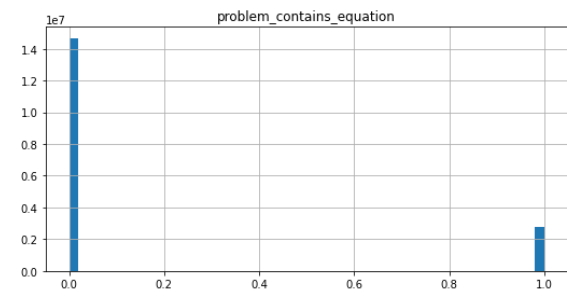
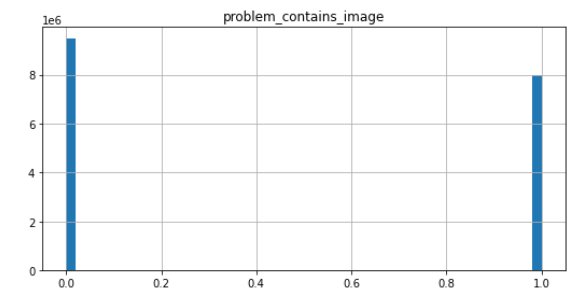
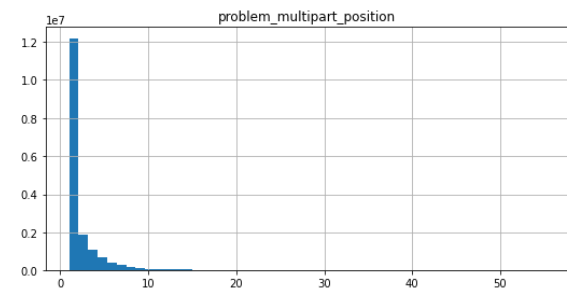
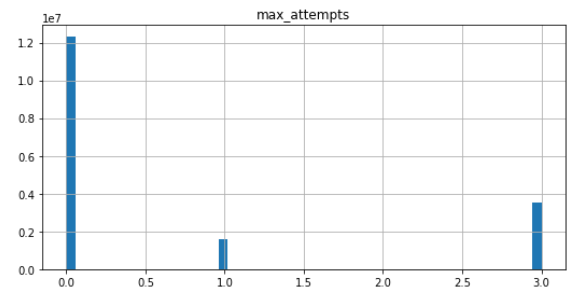
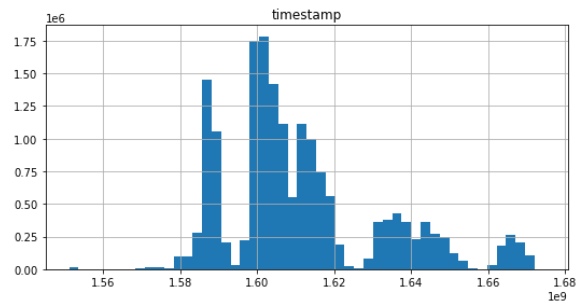
Handling Missing Data

1. Removing Variables with Excessive Missing Data: The 'problem_text_bert_pca' has a high percentage of missing data (70.52%). Since it has a large proportion of missing values, it can negatively impact the model's performance, and we decided to remove it from our dataset.
2. Incorporating 'score_viewable' and 'continuous_score_viewable' into 'action': As mentioned earlier, we have included the 'score_viewable' and 'continuous_score_viewable' into the 'action' feature since their values are only present for the 'problem_started' action. By combining these variables, we can better handle the missing data and capture more information about student behavior.

After cleaning data, we also check the details for each data that help us get more information about these:

1. We removed the 'problem_skill_description' variable due to its high cardinality. Visualizing the variable's distribution using histograms showed that it contains too many unique values, which may negatively impact the model's performance.





	score
score	1.000000
timestamp	-0.085844
max_attempts	0.100890
problem_contains_image	0.096621
problem_contains_equation	-0.075940
problem_contains_video	-0.004244
action_answer_requested	-0.238309
action_correct_response	0.187400
action_explanation_requested	-0.063167
action_hint_requested	-0.080088
action_live_tutor_requested	0.004117
action_open_response	-0.017713
action_problem_finished	0.206741
action_problem_started_not_viewable_not_continuous	-0.037008
action_problem_started_viewable_continuous	0.105122
action_problem_started_viewable_not_continuous	-0.036518
action_skill_related_video_requested	-0.007574
action_wrong_response	-0.184274
problem_type_Algebraic Expression	0.026330
problem_type_Check All That Apply	-0.030686
problem_type_Exact Fraction	-0.022129
problem_type_Exact Match (case sensitive)	0.013164
problem_type_Exact Match (ignore case)	-0.022462
problem_type_Multiple Choice	-0.025470
problem_type_Number	0.049662
problem_type_Numeric Expression	-0.052064
problem_type_Ordering	-0.010741
problem_type_Ungraded Open Response	-0.034460
action_count	0.036442

Then, we also need to handling Categorical Variables which is converting categorical variables, such as problem_type into numerical form using methods 'get_dummies()' function. This transformation allows machine learning algorithms to process the data more efficiently.

#	Column	Data Type
0	id	int64
1	assignment_log_id	object
2	problem_id	object
3	score	int64
4	timestamp	float64
5	max_attempts	float64
6	problem_contains_image	float64
7	problem_contains_equation	float64
8	problem_contains_video	float64
9	action_answer_requested	float64
10	action_correct_response	float64
11	action_explanation_requested	float64
12	action_hint_requested	float64
13	action_live_tutor_requested	float64
14	action_open_response	float64
15	action_problem_finished	float64
16	action_problem_started_not_viewable_not_continuous	float64
17	action_problem_started_viewable_continuous	float64
18	action_problem_started_viewable_not_continuous	float64
...

Totally 32 columns

Models used to model behavior

Model	AUC	F-1	RMSE	AUC in Kaggle
Linear Regression Model	0.65	0.74	0.59	0.61
Logical Regression Model	0.58	0.43	0.64	0.51
Random Forest Model	0.69	0.68	0.56	0.51

Various models can be used to predict the binary measure of problem correctness. In this analysis, we use Linear Regression, Logistic Regression, and Decision Tree models for this task.

1. Linear Regression Model

- Advantages:

- Simplicity: Linear regression is easy to understand and implement, making it a good starting point for modeling relationships between variables.
- Interpretability: The model provides interpretable coefficients that indicate the direction and strength of the relationship between the predictor variables and the outcome.
- Computationally Efficient: Linear regression is computationally inexpensive, making it suitable for large datasets.

- Disadvantages:

- Limited Complexity: Linear regression assumes a linear relationship between the predictor variables and the outcome, which may not always hold true in real-world scenarios.
- Inappropriate for Binary Outcomes: Linear regression is designed for continuous outcomes and may not be the best choice for predicting binary outcomes like problem correctness.

2. Logistic Regression Model

- **Advantages:**
 - **Suitable for Binary Outcomes:** Logistic regression is specifically designed for predicting binary outcomes, making it more appropriate for the prediction task in this competition.
 - **Interpretability:** Like linear regression, logistic regression provides interpretable coefficients that represent the relationship between predictor variables and the binary outcome.
 - **Robust to Small Feature Changes:** The model is relatively stable and robust to small changes in the features.
- **Disadvantages:**
 - **Assumes Linearity:** Logistic regression also assumes a linear relationship between the logit-transformed outcome and predictor variables, which may not always be the case.
 - **Sensitive to Outliers:** Logistic regression can be sensitive to outliers in the data, which can impact the model's performance.

3. Random Forest Model

- **Advantages:**
 - **Handles Non-linear Relationships:** Random Forest can model complex, non-linear relationships, making it more flexible than linear and logistic regression models.
 - **Robust to Overfitting:** Random Forests reduce the risk of overfitting by averaging the predictions of multiple trees, leading to better generalization to unseen data.
- **Disadvantages:**
 - **Model Complexity:** Random Forest models are more complex and harder to interpret compared to linear and logistic regression models.
 - **Computationally Intensive:** Random Forest can be computationally expensive, particularly for large datasets and a large number of trees.
 - **Less Interpretable:** The predictions generated by a Random Forest model are less interpretable, making it more challenging to explain the model's decisions.

Results and interpretation

According to the Table from Models used to model behavior, we get the highest accuracy in Linear Regression Model with 0.61. After implementing various models to predict student

performance on individual problems, we will now discuss the results and interpret the findings. We will also suggest potential improvements and alternative approaches for future iterations.

1. **Unexpected Outcome for LDA:** The highest accuracy achieved with the LDA model was unexpected. This could be due to our feature engineering process, where we might not have considered all relevant variables. The need to remove multi-part variables due to computational constraints may have contributed to suboptimal results.
2. **Handling Categorical Variables:** We used the 'get_dummies' function to handle categorical variables, which might not be the best choice. An alternative approach, such as using sklearn's OneHotEncoder, could potentially yield better results.
3. **Data Splitting:** We split the data using a 70/30 ratio, but other splitting methods like random split and stratified sampling split can be explored to see if they lead to improved performance.
4. **Model Tuning:** We can try various model tuning techniques, such as cross-validation and grid search, to optimize the model parameters and potentially improve the prediction accuracy.
5. **Exploring Additional Models:** We can also experiment with other models, such as Knowledge Tracing, to assess their effectiveness in predicting student performance on individual problems.
6. **NULL values:** In the final step of using the model for student performance predictions, we encountered 11 NULL values. Directly replacing NULL values with 0 might not be the best approach, as it could negatively impact the model's performance.

The results obtained from our current models indicate that there is room for improvement. By refining the feature engineering process, exploring alternative approaches for handling categorical variables, adjusting data splitting methods, tuning models, and experimenting with additional models, we can potentially enhance the prediction accuracy and better understand the factors influencing student performance on individual problems.