# Modeling NAEP Test-Taking Behavior Using Educational Process Analysis

Nirmal Patel
Playpower Labs
nirmal@playpowerlabs.com

Aditya Sharma
Playpower Labs
aditya@playpowerlabs.com

Tirth Shah
Playpower Labs
tirth.shah@playpowerlabs.com

Derek Lomas
Delft University of Technology
j.d.lomas@tudelft.nl

Process Analysis is an emerging approach to discover meaningful knowledge from temporal educational data. The study presented in this paper shows how we used Process Analysis methods on the National Assessment of Educational Progress (NAEP) test data for modeling and predicting student test-taking behavior. Our process-oriented data exploration gave us insightful findings of how students were interacting with the digital assessment system over time. To discover what processes students were following during the NAEP Digital Assessment, we first developed an innovative set of research questions. Then, we used Process Analysis methods to answer these questions and created a set of features that described student behavior over time. These features were used to create an ensemble model that aimed to accurately predict the digital test-taking efficiency of the students taking NAEP. Our model emerged as one of the most successful models in the 2019 NAEP Data Mining Competition, scoring second place out of 89 teams.

**Keywords:** process analysis, behavior modeling, test-taking behavior, curriculum pacing, process mining

## 1. INTRODUCTION

Decades have passed since researchers first started using computers to deliver educational experiences. Since the time of PLATO, one of the first digital instructional programs, researchers have been fascinated by the potential of student data collection and analysis (Slattow, 1977, p.145). Today's digital learning platforms collect vast amounts of student activity data and then use these data to inform the student experience (e.g., recommendations and resources). Educational data are also used by teachers to understand their students and their needs (Data Quality Campaign, 2018). We can also use educational data to build models that can predict future student behavior. Our study in this paper describes how we used data from the US Government's National Assessment of Educational Progress[1] (NAEP) test to develop a novel student behavior detection model. The model was designed to predict, as much as 20 minutes in advance, how the student was going to spend their time during the NAEP Digital Assessment.

[1] https://nces.ed.gov/nationsreportcard/

Our work was conducted as a part of the 2019 NAEP Data Mining Competition (Baker et al., 2019) where competition organizers invited the participants to understand effective and ineffective test-taking behaviors in the NAEP Digital Assessment, and determine how quickly these behaviors could be detected. The ultimate goal of the competition was to build a model that could predict whether the student was going to spend their time in NAEP efficiently or not. To build our model, we extensively used Educational Process Analysis methods to design innovative features that aimed at accurately predicting student test-taking behavior. The use of Process Analysis methods gave our model predictive power and explainability; not only were we able to make accurate predictions, but we also gained insight into how students behaved during the test.

### 1.1. Educational Process Analysis

Educational Process Analysis aims to discover latent learning and teaching processes that are hidden within temporal educational data. Typically, the process data has information about what students and teachers do over time. From such data, we can discover different representations of the unobserved learning and teaches processes that might be producing the observed data. In other words, the process data can be thought of as resulting from a set of hidden learning and teaching processes occurring within students and teachers.

Processes hidden in the data can be represented by various types of constructs and models. Some examples are directed graphs, process model representations such as Petri Nets, Heuristic Nets, or Fuzzy Nets, graphical models such as Hidden Markov Models or Bayesian Networks, simpler constructs like Markov Chain Transition Matrices, or even trivial representations like discrete event sequences. Techniques such as Association Rule Mining (García et al., 2010), Sequential Pattern Mining (Zhou et al., 2010), Process Mining (Trčka et al., 2010; Bogarín et al., 2018), Graph-Based Analysis (Lynch et al., 2017; Patel et al., 2017), and Curriculum Pacing (Patel et al., 2018) can help us discover different representations of educational processes from the data. For example, the Rule Mining methods can discover which student/teacher interactions follow each other more frequently. Pattern Mining methods can reveal frequent sequences of actions in the data. Process Models and Graph-Based Analysis can give an end-to-end view of the student interaction data as process models or graphs, whereas the Curriculum Pacing method produces a clear visualization of how students follow the curriculum over time.

Educational process data can come in many different shapes and sizes, and we have to use different methods for different types of data. For example, to analyze task-level data with a low amount of variance, we can use graph-based algorithms or process modeling algorithms such as Heuristic Miner (Bogarín et al., 2018). These algorithms become difficult to use when there is a high amount of complexity in the data. This is often the case with click-stream data, where we can use algorithms like Fuzzy Miner (Bogarín et al., 2018) that give more flexibility with 'zooming in and out' of the process maps so that we can easily look at both more and less frequent behaviors. If the data have a high amount of variance, meaning that there are too many student learning processes or behaviors tied up with each other, we can use sequence clustering methods to group student data with similar temporal features and analyze them separately (Bogarín et al., 2014; Patel et al., 2017).

### 1.2. Paper Organization

The rest of the paper is structured as follows: Section 2 reviews the earlier work on behavior detection, focusing on which methods and approaches were used to build the behavior detection models. Section 3 briefly describes the NAEP Data Mining Challenge and the behavior detection model that the participants had to build. Section 4 outlines our exploratory analysis of the data and presents a process view of the student data. Section 5 describes our feature engineering work, while Sections 6 and 7 detail the modeling approach and final results. Section 8 gives details about the open-source code that readers can download to reproduce our results. The remainder of the paper contains a discussion and some ideas for future analyses.

## 2. Prior Work

A number of studies have been published around predicting student behavior using educational data. Many of these studies have also correlated student behavior with students' learning outcomes.

Behavior detection models are specifically designed to infer whether a student is engaged in a specific type of complex behavior (Baker, 2015). These models can be used to help digital systems (or teachers) take appropriate actions. This type of data-informed learning experience can possibly lead to better outcomes for the students. A recent study by Holstein et al. showed that when teachers were given real-time data about student knowledge and behavior, student learning was impacted positively (Holstein et al., 2018). Besides detecting complex student behaviors in advance, we can also study their relationships with student learning. There are several classes of behaviors in the digital learning systems that researchers have detected using automated models. Some of the examples are listed below (Baker, 2015):

- Gaming the System Behavior: Students trying to figure out how they can succeed in going through the digital learning system without learning e.g. clicking the next button repeatedly, or getting the hints without trying to solve the problems
- Off-Task Behavior: Students pausing during their interaction with the learning system for some external reason
- Carelessness Behavior: Students making errors that are not due to their lack of knowledge
- Without Thinking Fastidiously Behavior: Students doing the actions in the learning system that are not related to the learning task e.g. repeatedly changing their avatar, piling up virtual currency by taking the same assessment again and again
- Help Avoidance behavior: Students not asking for help in the digital system, even if they need it

The analysis presented in Baker et al. (2004) showed that gaming the system behavior was negatively correlated with the learning outcomes. The behavior detection model in Baker et al. (2004) used 24 features created from the student data within a latent response model to predict the frequency of the gaming the system behavior. In Baker et al. (2006), researchers devised a system that prevented students from gaming by giving them more exercises in the areas where they showed the gaming behavior. This preventive system reduced the gaming behavior of the

students, and the students who received more resources as a result of their gaming behavior performed better in comparison to the students who did not receive additional resources. A much bigger model to detect the gaming behavior was presented in Walonoski and Heffernan (2006) where the researchers used an ensemble learning approach. Their final model was an ensemble of 12 different algorithms which included Decision Tree methods, methods such as K-Nearest Neighbors, Locally Weighted Learning methods, Bayesian methods, a Neural Network, a Propositional-Logic Rule Learning Algorithm and a Logistic Regression model. They used a total of 1430 features that were mined from student log data. We took a similar ensemble approach in this paper, but our set of features remained significantly smaller.

Off-Task Behavior was detected by Baker (2007) using a latent response model and a combination of features from several different sources. Another study to model the off-task behavior used Least Squares and Ridge Regression, and utilized a set of time features, performance features, and mouse movement features to detect the student behavior (Cetintas et al., 2009). Carelessness Behavior was modeled in Pedro et al. (2011) by using Contextual-Slip-and-Guess Estimation based on Bayesian Knowledge Tracing. Another study that used the same method found that the students who had high levels of concentration demonstrated the most careless behavior (Pedro et al., 2011). A study that attempted to model the Without Thinking Fastidiously behavior tested 11 common classification algorithms such as Naive Bayes, J48 Decision Trees, etc. on the student data (Wixon et al., 2012). The best model performance was achieved by the Partial Decision Tree algorithm. Results presented in Rowe et al. (2009) found that student activity in the digital tutor that was not related to learning was negatively correlated with the pre and post-test scores of classroom students. Help Avoidance behavior was detected in Aleven and Koedinger (2000) and Aleven et al. (2006) by creating more production rules within the intelligent tutor.

Behavior detection has also been studied in the context of digital testing environments. Many studies have investigated how student response times to different test items relate to the outcomes of interest. In Şahin and Colvin (2020), researchers used item response rapidness to detect disengaged behavior. Response rapidness was also used to identify differences in the item correctness in Goldhammer et al. (2016) where the researchers found (p. 15, ibid) that a small proportion of students responding to an item very rapidly had close to zero problem correctness. This 'clicking through the test items' behavior can also be exhibited by high performing students who are simply bored or not interested in taking the online test. Guo et al. (2016) proposed that such zero correctness item responses that are correlated with very low response times should be removed from the item and test validation process because they will have undesirable impact on the measurement procedures. Several other studies have used response times of the students to understand and predict the outcomes of the digital tests (Kong et al., 2007; Lee and Jia, 2014). In this study, we use a near identical measure of how rapidly students interact with a test item to predict the target behavior. Şahin (2019) used Latent Profile Analysis to look at how students allocated times to different test items in the NAEP. They discovered 4 distinct groups in their sample that they described as 1) little time on first (problem) visit, 2) balanced time (across problem visits and revisits), 3) little revisit with more time at the end, and 4) little revisit with less time at the end. The researchers found that these four groups differed in the average outcomes with group 4 scoring the lowest (p. 21, ibid).

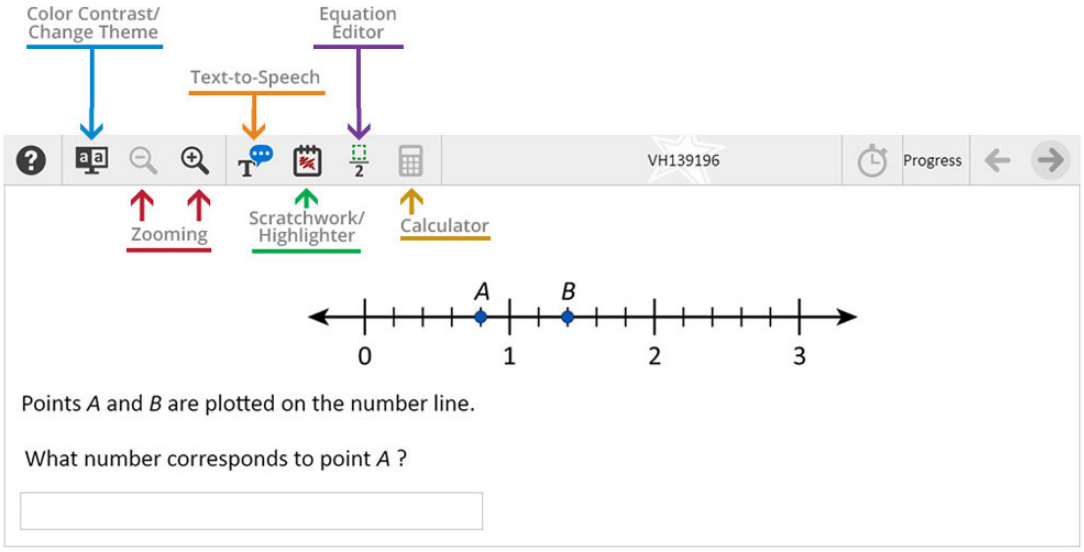


Figure 1: User interface of the NAEP digital test.

## 3. NAEP Data Mining Challenge

In 2019, Baker et al. announced the NAEP Data Mining challenge that aimed at engaging the participants in modeling the student test-taking behavior in the NAEP Digital Assessment (Baker et al., 2019). The broader goal of the competition was to develop metrics for measuring students' test-taking activities, better understand effective and ineffective test-taking behaviors, and determine how quickly these behaviors could be detected. The organizers of the competition provided NAEP Process Data[2] for around 2400 students who took the NAEP test in early 2017. Researchers have proposed various ways to leverage NAEP Process Data to predict outcomes of interest (Bergner and von Davier, 2019). The organizers asked the competition participants to use the process data to build a behavior detection model that predicted whether the students were going to spend their time in the test efficiently or not.

The NAEP test is used by the US government to measure student achievement across the country. The digital version of the test collects detailed data about the student test-taking process. The test is administered in a variety of subjects, and the questions used in the previous assessment are also available publicly[3]. For the competition, we received the process data from the Math test. The data contained details about how students interacted with the user interface of the digital test over time. Figure 1 shows how the UI looked while the students took the test. Apart from seeing the question in the UI, students were also given several learning aids that they could use to solve the problem. The data about how students used these learning aids was adequately captured and given to the competition participants.

### 3.1. Competition Structure and the Outcome Variable

The NAEP Math Test consisted of two 30 minute blocks of time called Block A and Block B. In both blocks, students were given a fixed set of questions. The task of the competition was to use

[2] https://www.nationsreportcard.gov/process_data/
[3] https://nces.ed.gov/nationsreportcard/nqt/

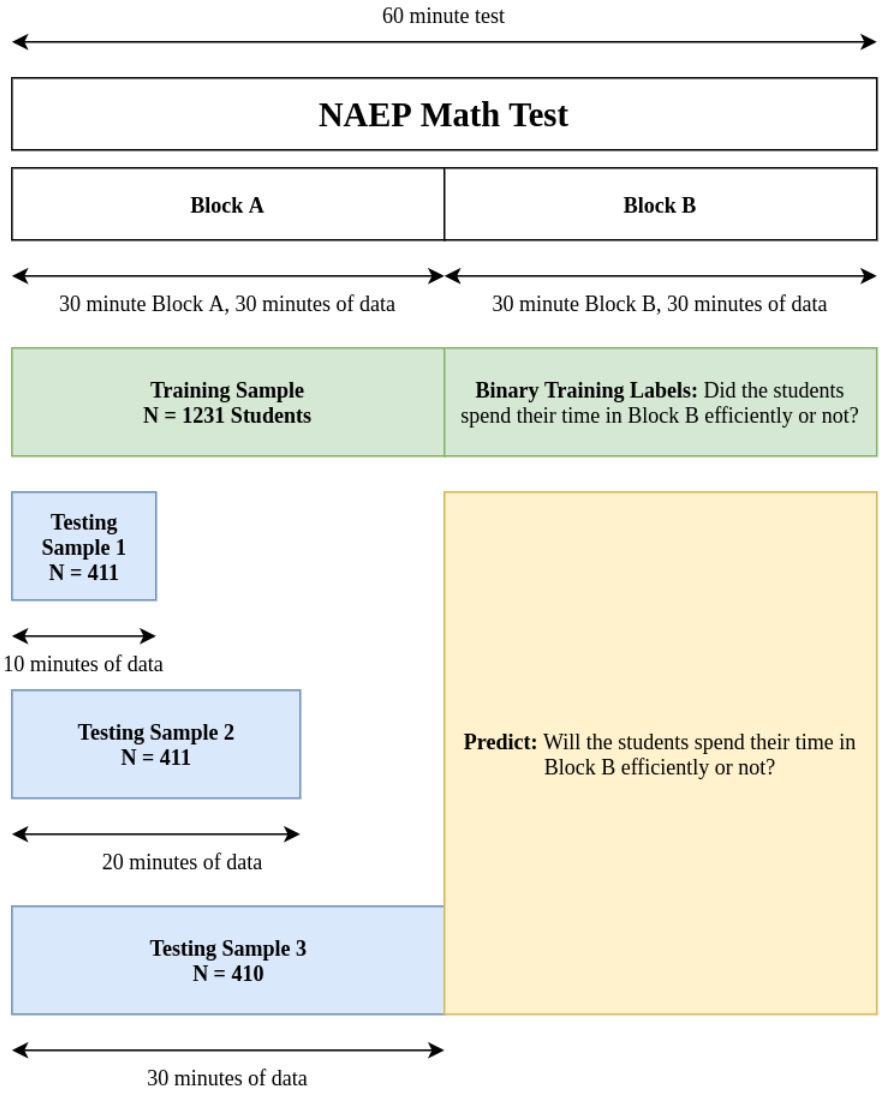


Figure 2: Structure of the 2019 NAEP Data Mining Challenge.

data from Block A to predict student behavior in Block B. The target variable to be predicted was a binary indicator of whether or not the student would spent their time in Block B efficiently. The organizers of the competition used a method described in Patikorn and Heffernan (2019) to define what the efficiency meant and they generated the outcome labels of 'efficient' and 'inefficient' accordingly[4]. Both the training and test set of the data were from Block A, while the outcome labels were from Block B.

The testing data was organized slightly differently from the training data. While the training data had full 30 minute of process data for every student, the testing data was divided in 3 different subgroups of 10, 20, and 30 minutes. Each of these subgroups had the respective minutes of

[4]"We defined efficient usage of time as 1) being able to complete all problems in Block B, and 2) being able to allocate a reasonable amount of time to solve each problem...[namely] for each problem in Block B, we ranked the total amount of time each student took to complete each problem, and used the 5th percentile as the cut-off for the "reasonable amount of time." (Patikorn and Heffernan, 2019)

Table 1: Columns of the NAEP Process Data.

| Variable | Description |
|---|---|
| STUDENTID | Unique identifier of the student |
| Block | Block of the NAEP test, A or B |
| AccessionNumber | Unique identifier of the question that the student is attempting |
| ItemType | Type of the question e.g. MCQ, Fill in the Blank, etc. |
| Observable | The name of the action that student took,<br>e.g. clicking, dragging, scrolling, typing, opening a calculator |
| ExtendedInfo | Metadata of the student action in a JSON format<br>e.g. how much did the student scroll, what key she pressed<br>on the calculator, what digit she typed as a response, etc. |
| EventTime | Time when the student interaction occurred |

data for the students within them. This was done to find out how early we can detect the target behavior in Block B. For example, if we can accurately detect student behavior in Block B after the first 10 minutes of Block A, the student can be given an appropriate intervention. Figure 2 shows the structure of the competition more clearly.

Due to privacy and test validity reasons, additional data about student performance and demographics were not available. For instance, we were not provided with details about student scores on individual items nor overall scores on the test. The only data available for use were those listed in Table 1.

### 3.2. Model Evaluation

The competition organizers created an online submission form where we had to submit our predictions for the 1,232 students in the test set. Participants received a feedback based on a random sample of the test data which acted as a public validation set. The actual test set was held back for the final results of the competition. Adjusted AUC and Adjusted Kappa values for the public validation set were provided once every day to the participants. These metrics are described more in Section 7.1. This once-a-day mechanism prevented participants from overfitting their models to the public validation set. The evaluation of the submissions on the private test set was made public when the final competition results were announced.

## 4. Exploratory Data Analysis

We started our model-building process by extensively analyzing the process data that were provided to us. The data consisted of 1.19 million observations for 2,463 students across both the training (N = 1,231) and the test set (N = 1,232). The data from the sample were spread out over February and March of 2017. The student data were in an event log format, where each student interaction in the digital assessment system resulted in one observation in the dataset. Each observation had 7 different variables. They are listed in Table 1. It is a very typical practice to format event logs as an actor-verb-object triplets. The data sample had student ID for the actor part, multiple variables describing the type and subtype of student action which was the verb part, and the question ID and the related metadata for the object part. Figure 3 shows a random sample of 10 rows from the dataset.

| | STUDENTID | Block | AccessionNumber | ItemType | Observable | ExtendedInfo | EventTime |
|---|---|---|---|---|---|---|---|
| 1 | 2333226481 | A | VH134366 | MultipleFillInBlank | Math Keypress | {"numericIdentifier":"4","partId":"","contentMathML... | 2017-02-01 20:48:27 |
| 2 | 2333128170 | A | VH098753 | MCSS | Move Calculator | TI30 | 2017-02-09 18:49:53 |
| 3 | 2333325994 | A | VH134366 | MultipleFillInBlank | Receive Focus | 2 | 2017-02-08 16:02:49 |
| 4 | 2333239435 | A | VH134387 | FillInBlank | Math Keypress | {"numericIdentifier":"1","partId":"","contentMathML... | 2017-03-07 17:32:54 |
| 5 | 2333177910 | A | VH098839 | MCSS | Move Calculator | TI30 | 2017-03-08 15:02:58 |
| 6 | 2333347017 | A | VH134366 | MultipleFillInBlank | Receive Focus | 3 | 2017-03-13 16:19:40 |
| 7 | 2333338930 | A | VH134387 | FillInBlank | Receive Focus | 1 | 2017-03-06 17:36:19 |
| 8 | 2333349842 | A | VH134366 | MultipleFillInBlank | Receive Focus | 2 | 2017-02-09 13:35:52 |
| 9 | 2333372071 | A | VH139196 | CompositeCR | First Text Change | B,1 | 2017-02-08 18:24:40 |
| 10 | 2333081475 | A | VH139196 | CompositeCR | Receive Focus | Part A, 1 | 2017-02-15 18:04:49 |

Figure 3: A sample of the NAEP Process Data.

The distribution of item types in the dataset was quite skewed, with the MCQ (Multiple Choice Question) type having 14 items, Fill in the Blank having 2 items, and all of the other question types having just 1 item (the other types were Matching, Multiple Fill in the Blank, and Composite Constructed Response). For each of the question types, there were several possible actions. In total, there were 42 unique actions in the data. Some actions were only present in some of the question types. Figure 4 gives a clear picture of which actions were frequent in which questions. Looking closely at the plot, we can see that the frequency of the Open Calculator was different in different questions, that there were some actions that were not common across all of the item types, that the Draw action was done with different frequency across the problems, etc. We found that both the calculator and drawing facilities were used more in the questions where students had to apply procedural knowledge rather than declarative knowledge. For example, in question VH098810[5], students simply had to answer with the correct unit so their calculator and draw pad usage was quite low. By comparison, question VH098519[6] was a geometry question, and students used both the calculator and drawing considerably more. Our data exploration was aimed at discovering differences of this nature and seeing whether any of them correlated with the outcome variable.

### 4.1. Process Analysis

The outcome variable that we were trying to predict was whether the student was going to spend their time in the NAEP test efficiently or not. Thinking about the predictor variables that would allow us to predict the outcome correctly, we felt that different aspects of the student's problem-solving process might act as a good set of predictors, because the process in which the student was taking part was a construct that was likely aligned with the behavioral outcome. In other words, we hoped that if we had behavioral predictors, they would be better at predicting a behavioral outcome. Ultimately, though, all of our hypotheses were validated by the feedback that our predictive model received on the public validation set.

Given the smaller sample size, we took a feature based approach to build our predictive model. We first hypothesized the different types of behaviors that might predict the outcome, and then we converted those behaviors into features. Our process to build the model was as

[5] https://cotw.naep.ed.gov/student/grade8/MAT/VH098810/toolbarOn
[6] https://cotw.naep.ed.gov/student/grade8/MAT/VH098519/toolbarOn

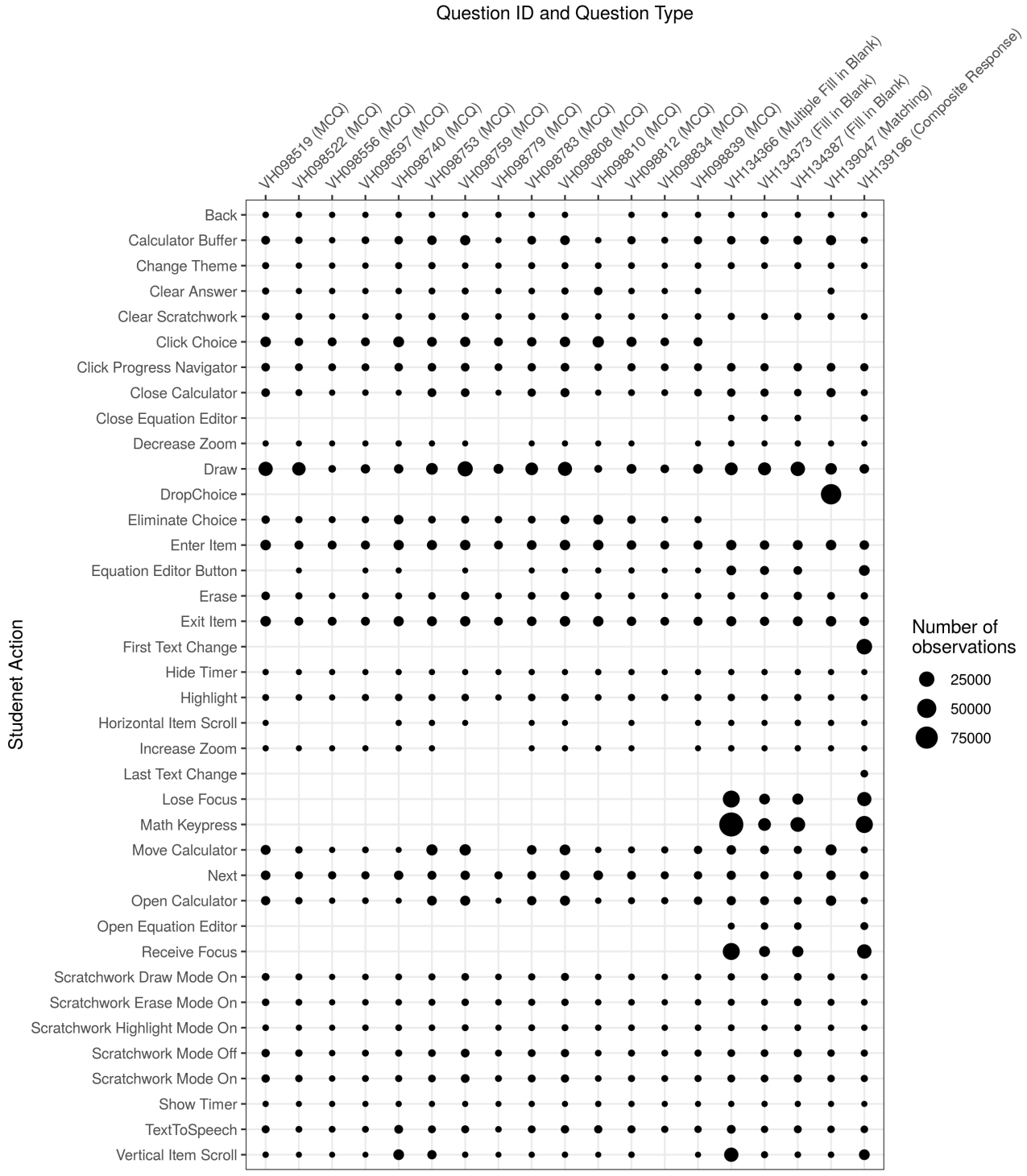


Figure 4: Questions and the possible actions within them.

follows:

1. Come up with a list of innovative research questions about student test-taking process
2. Turn those research questions into features
3. See if the new features are helpful in increasing the predictive accuracy
4. Continue the same process until the end of the competition

Time is a critical factor in building the predictive models. We could only continue to follow our process until the end of the competition, even though on the last day of the competition, we had more ideas to try. Below is our list of research questions that we put together during the course of the competition, along with our rationales behind them:

- *How did the students go through the questions over time?* The pattern in which students took the first half of the test can reveal whether they were going to be effective or ineffective during the later half of the test.
- *How much time did the students spend on each problem?* The distribution of the time spent on different problems can tell us how well balanced it was. If a student spent a lot of time on one problem and did not solve other problems, this was most likely an indicator of inefficient behavior.
- *What was the distribution of the time gaps between student actions?* The rhythm profile of the student actions can tell us something about the effectiveness of their test-taking process. Maybe there was an 'ideal rhythm' and large distance from that rhythm was a likely indicator of inefficiency.
- *How quickly did the students enter an answer after seeing the problem?* If students acted quickly after seeing the questions, they might be trying to click through the test without really attempting the questions, or being hasty. Both of these phenomena could indicate an inefficient test-taking approach.
- *How many times did the students switch between the problems?* If a student had to switch between problems many times, it was likely that they had some issue during the test.
- *What was the problem completion rate?* If a student had a low problem completion rate, it was likely that they were not spending their time efficiently during the test.
- *What was the student accuracy?* Maybe more accurate students were more efficient.
- *How many times different actions occurred during the test?* Counts of actions (along with averages, minimums, maximums, and standard deviations) are very typical predictors in educational data, so we included them without any further thinking.

We found that process-oriented data analysis presented us with novel lenses to look at the data and generate better questions around what type of student interaction patterns might predict the student test-taking behavior. There are several approaches available to take a more process-oriented look at the data. Two of the most distinct approaches that we used were Curriculum Pacing and Process Mining. They are described below.

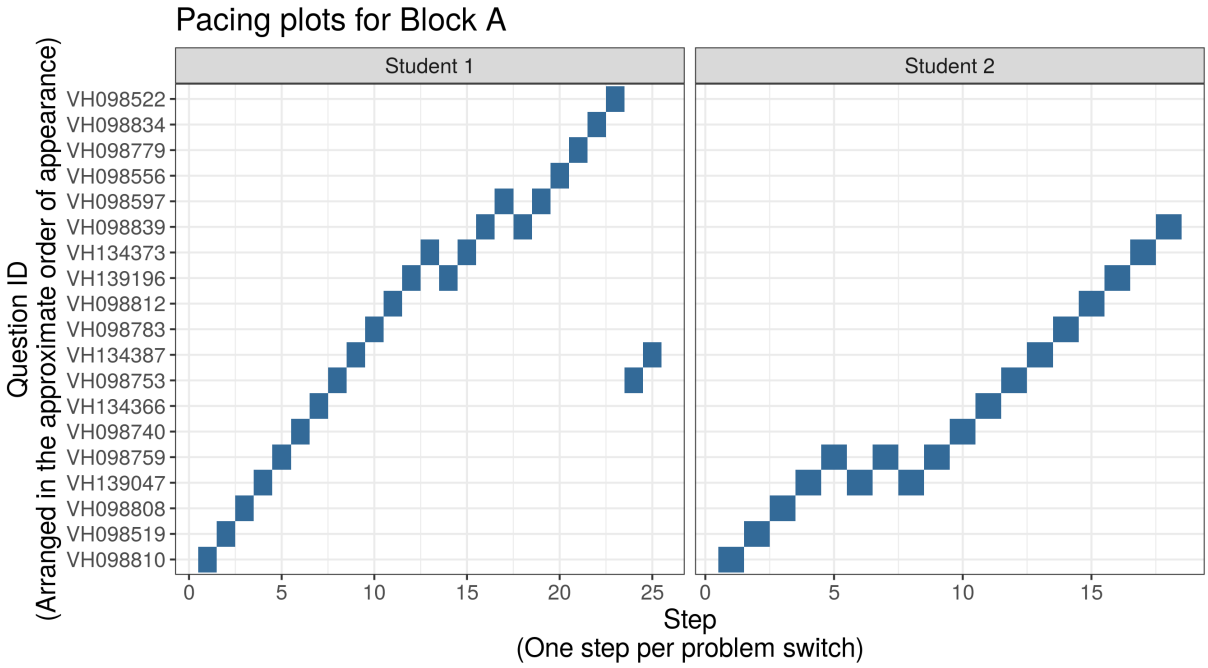


Figure 5: Example Pacing Plots of two randomly selected students. The X axis of the plot represents student steps which occur when the students change the problem that they are working on. The Y axis of the plot has question IDs of the Block A of the test, arranged in their approximate order of appearance in the test.

#### 4.1.1. Curriculum Pacing

Curriculum Pacing Plots are a type of data educational data visualization developed to reveal the implementation of blended learning programs in classrooms (Patel et al., 2018). These visualizations allow us to look at how students progress over through a set of educational resources over time. Inherent to the plots is the set sequence of the learning materials that the students are expected to follow. This is a more common case in classroom instruction than in digital systems where students are often free to explore and can receive adaptive learning material. We adapted the Pacing Plots to look at how students stepped through the problems in the NAEP test over time. We first arranged the problems by their average row number within each student sample, and gave each problem an approximate 'problem number.' This number was simply the average of the problem's row number across all of the students. Then, for each student, we created an ordered list of problems that they attempted over time. The combination of this information allowed us to create Pacing Plots for the NAEP test. Figure 5 shows example Pacing Plots for two students' 30 minutes in Block A of the test.

We can see here that the student on the left went through all of the problems, and then came back to a couple of problems in the end. This was not the case for the student on the right. This student started off, then swung around two problems a few times, and then ended up not going all the way through the questions. Patel et al. took a large sample of their Pacing Plots and ran a clustering algorithm over these plots to group the similar ones and look at them separately. Doing so was not possible on our sample because of the smaller sample size. But if the pacing analysis is done on a larger sample of the NAEP test data, clustering can reveal quite interesting

groups of patterns. Figures 6 and 7 show two more sets of randomly selected 25 pacing plots, one for each of the outcome value (we have omitted the Y axis values in these plots, which are same as the Y axis values in Figure 5).

In these larger samples, we can see many interesting patterns! A very typical pattern appears for students 3, 22, and 27, who went through all of the problems twice. Student 17 went through all of the problems twice, but she did her first round of problems in a much fewer number of steps. We can see that many students (e.g. Student 50) went through all of the problems without ever going back and forth. We called this pattern a 'lockstep' pattern. We found that the prevalence of this exact pattern was not significantly different across the two outcome groups ($p = 0.27$). We can also see that many students kept switching between adjacent problems throughout the whole time. For examples, see students 10, 18, 23, 36, and 41. There are some students, e.g. 10 and 23, who could not make it all the way to the end. Some students had a mind of their own and had very unique ways of going through the problems. For example, Student 4 went on to do many of the problems in the reverse order! Some of these actions might be taken for exploration or review of the test before actually beginning to solve the problems.

If we can group these patterns based on a clustering or pattern detection approach, we can easily correlate them with various indicators of interest. Since we could not group pacing charts and give them labels automatically, we generated features from the pacing charts. For example, we divided the student event logs into 5 different time slots, and took averages of several indicators for each of the time slots. This gave us a numerical approximation of the visual information that we were seeing in the plots. Here is a concrete example. If for a given student there are 50 time steps and 10 problems, then we first divide the data in the 5 sequential segments each having 10 time steps. Then, within each segment, we can find the average order of the problem. This would give us a vector of length 5 which can numerically approximate the chart. Similarly, we can also take averages of any other indicators that we expect to vary in a certain fashion over time, and we would get back a vague numerical approximation of the 2D visual data.

#### 4.1.2. Process Mining

There were a few different ways to apply Process Mining (PM) algorithms to the NAEP data. We looked at the test level process maps that showed us how students proceeded through the question over time, and we also looked at how students interacted with the system within the question. To identify how the test-taking process of the efficient students was different from the inefficient students, we took the training data and divided it into two parts based on the outcome variable. Then, we looked at the process maps for each group separately. Our goal was to see if the two process maps of two outcome variables had significant differences. We defined 'significant' here as a visually significant difference between the process that could be discerned by examining them.

We used the Fuzzy Mining algorithm (Günther and Van Der Aalst, 2007) to build the process models. The modeling and visualizations were done within the `R` environment using the `fuzzymineR` package[7]. Fuzzy Mining algorithms take event data as input and return a set of metrics for nodes and edges of the process map that we can use to look at the process at different 'zoom levels'. For example, you can remove nodes from the process map that have a metric lower than a certain threshold value. The same can be done for the edges of the process map. In a nutshell, the visualizations of the process maps for the Fuzzy Mining algorithm are param-

[7] https://github.com/nirmalpatel/fuzzymineR

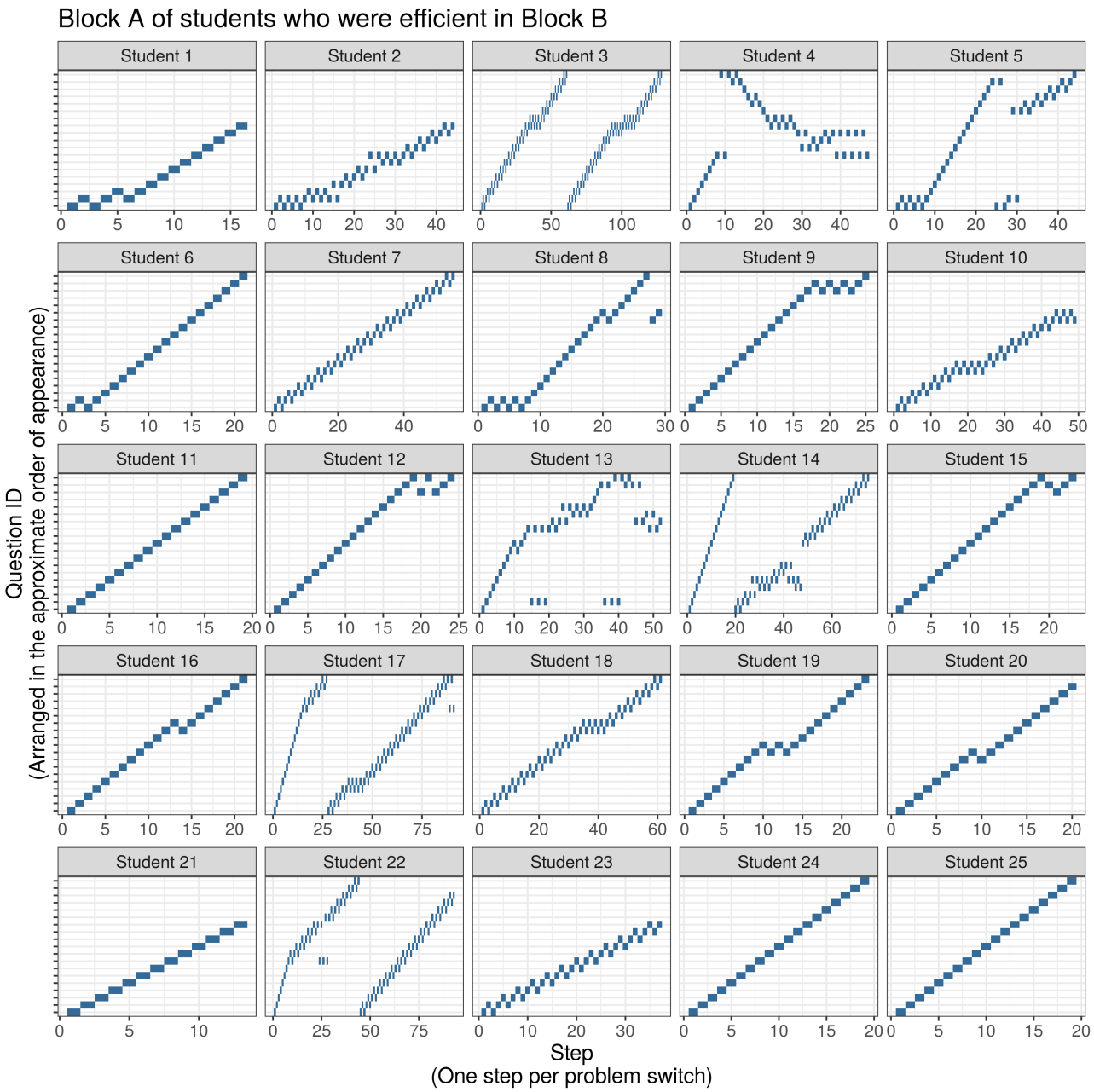


Figure 6: Block A Pacing Plots of students who were efficient in Block B. Y axis is the same as Figure 5.

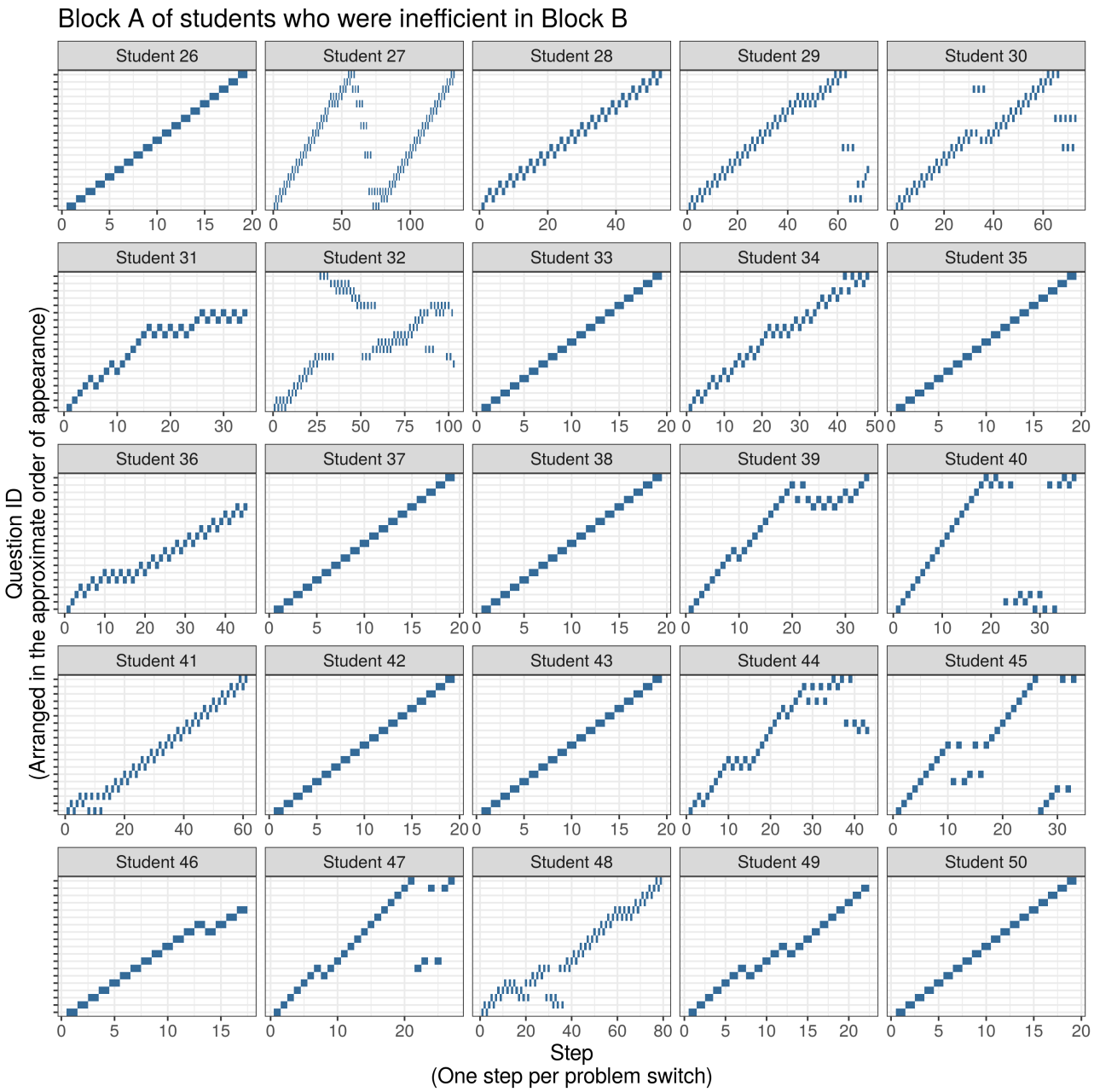


Figure 7: Block A Pacing Plots of students who were inefficient in Block B. Y axis is the same as Figure 5.

eterized, and changing values of those parameters will show or hide details of the process. We used a trial-and-error approach to find a good set of parameters that showed the most difference in the process maps, and then we examined those differences. We ensured that the parameters remained the same for both of the process maps (one for each of the outcome values).

The question level process map that we created had question IDs as nodes and the edges of the process map represented the transition between the problems. Two process maps, one for each of the outcome values, for the first 10 minutes of student data, are shown in Figure 8. In the process maps, the darker the nodes, the more frequently they are traversed in the process. Looking at these plots, we immediately identified a visual difference among them. The efficient group showed a strong path of students going through the problems from start to end (start and end are the very dark blue nodes on the left and right edges of the plots), while the inefficient students' path trailed off before the end of the 10 minutes.

We can see that in the top process map in Figure 8, there are some nodes that are not connected with the rest. This is because the edges that connect them with the rest of the process map are hidden due to a threshold parameter. When we add more edges to the process map, it quickly becomes complex and impossible to interpret. The process map in Figure 9 is not useful at all unless the researcher has obtained some special powers to make sense of it. For this reason, we used Fuzzy Mining primarily to explore the data and identify places where the student interaction processes for the different outcomes measures were sufficiently distinct. Once we identified those processes, we fit Markov Chain (MC) models to the student event log data where the states of the MC were the question IDs and the problem switches were modeled as change in the student state. MC matrices gave us a very simple way to convert process maps into numerical features that we could further utilize for creating features. Visualizations of the MC matrices also had visible differences. Figures 10 and 11 show the MC matrices for the efficient and inefficient students.

If we look closely, we can spot the difference between the two matrix visuals in Figures 10 and 11. There are differences present on the top, right, and bottom of the matrices. We called these matrices 'behavior prototypes,' meaning that they represented a prototypical student behavior. After creating these prototypes, we created a single MC transition matrix for each student. Then, we unrolled all of the matrices into vectors. This allowed us to calculate, for each student, her cosine similarity with each of the 'behavior prototypes.' The cosine similarities and their difference turned out be important (in one case, very important) features for the predictive models. In the given example, the cosine similarity between prototype matrices is 34 units, while their self similarities are 43 and 40 units. This tells us that the matrices are different not only visually but also numerically.

We can imagine doing the same exercise for the process models at the problem level where the states are the possible actions, e.g. Opening Calculator, Drawing, Answering, etc. Given the high dimensionality of the number of actions and more variance within student interactions, we found that the problem level process maps of the different outcomes were difficult to examine (see Figure 14). When we looked at the MC matrices for the question ID VH098759, we did not see many differences (see Figures 12 and 13).

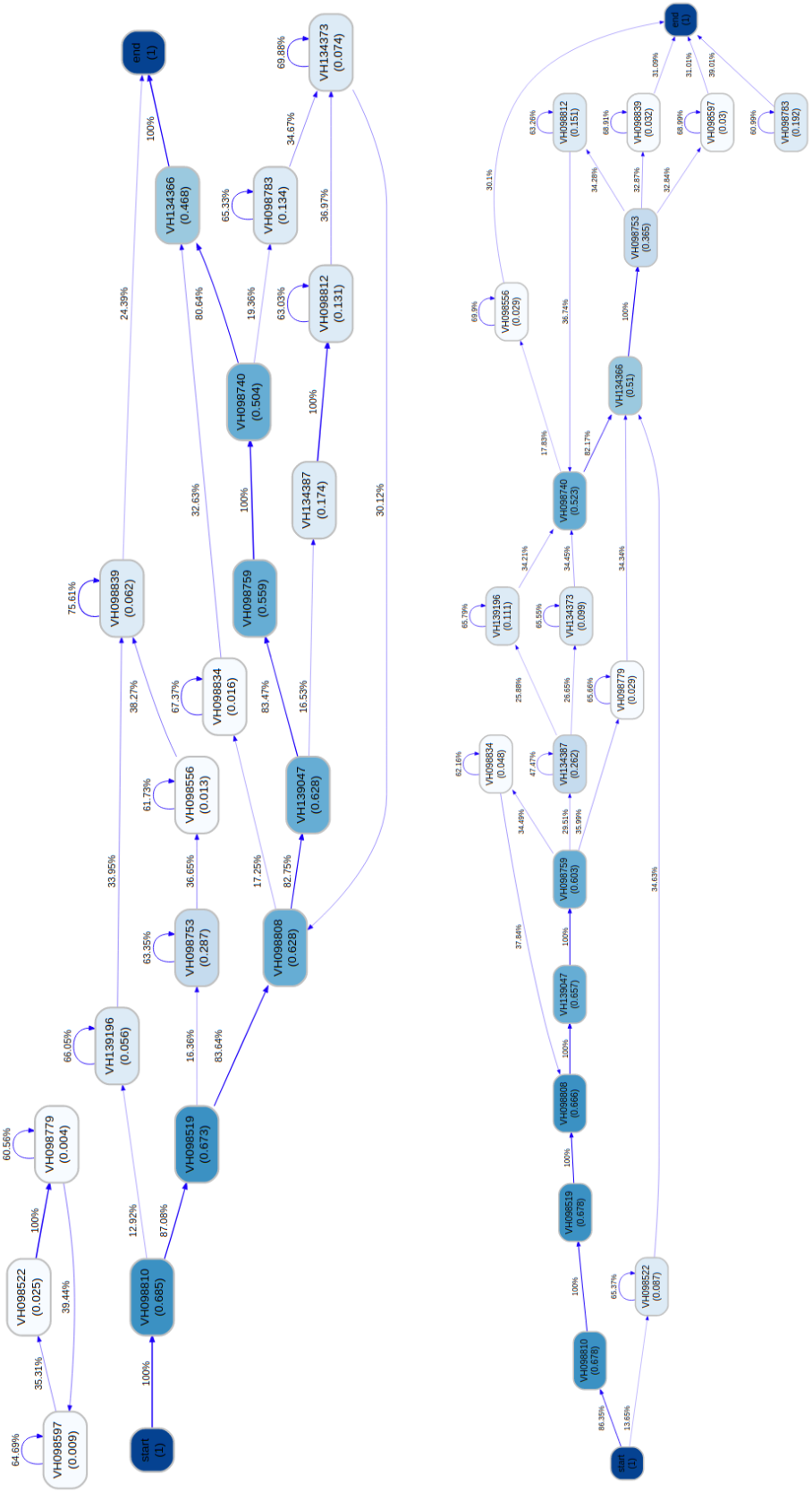


Figure 8: Process maps of efficient (top) and inefficient (bottom) students for the first 10 minutes of Block A. The nodes represent the questions and the edges represent students switching from one problem to another.

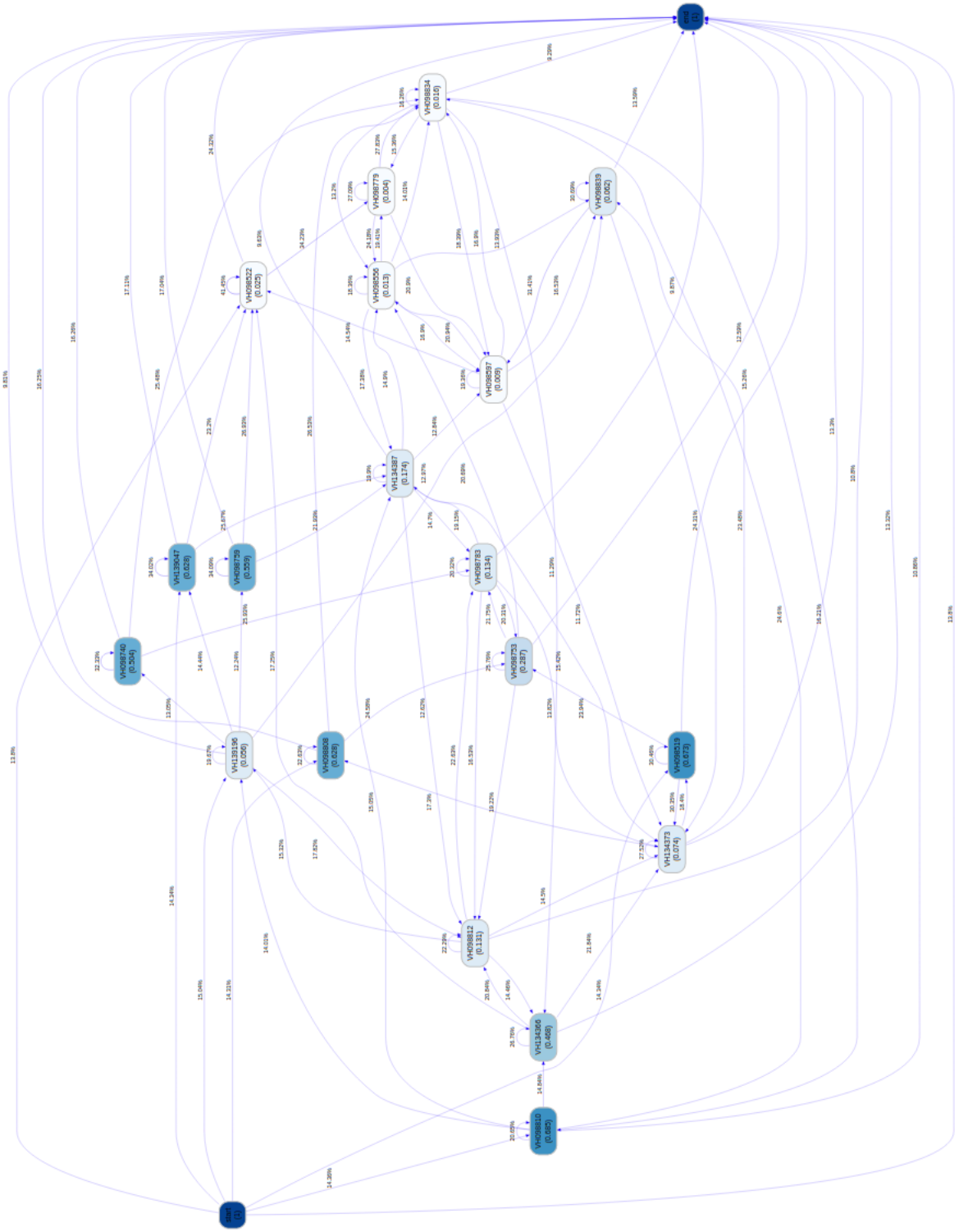

Figure 9: The top process map in Figure 8 with additional edges.

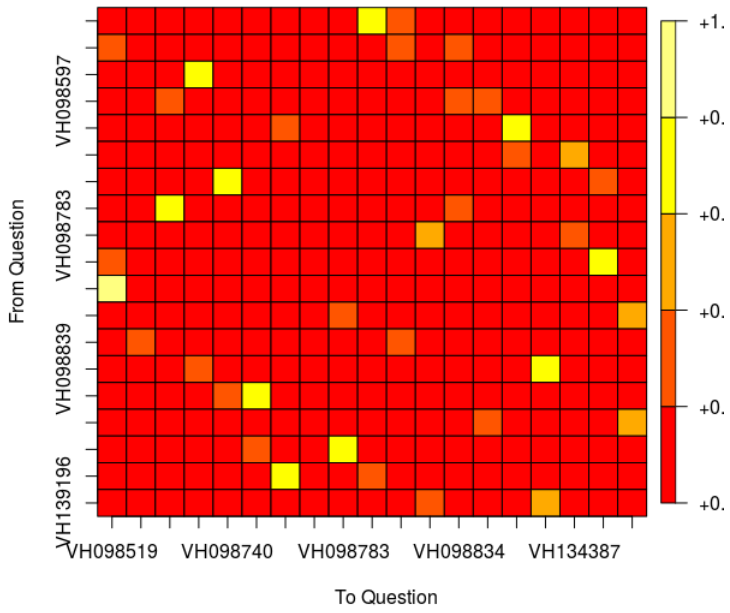


Figure 10: Markov Transition Matrix of the efficient students for the first 10 minutes of problem switches.

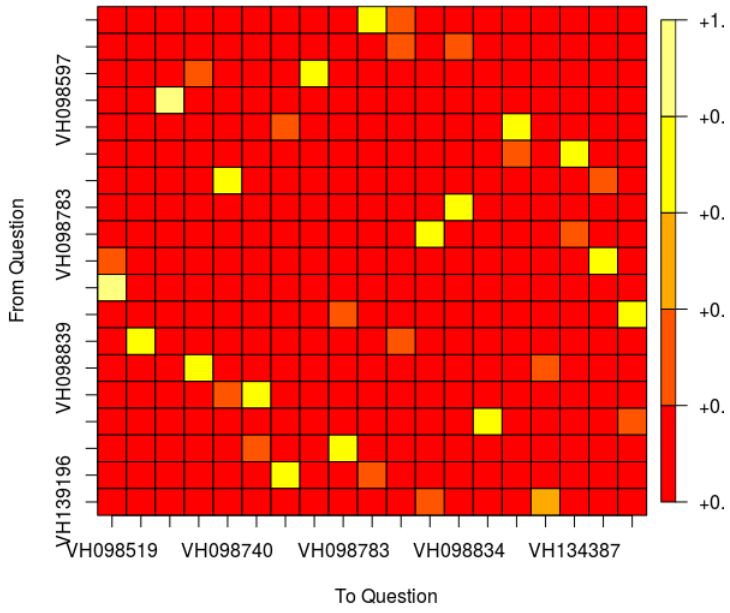


Figure 11: Markov Transition Matrix of the inefficient students for the first 10 minutes of problem switches.

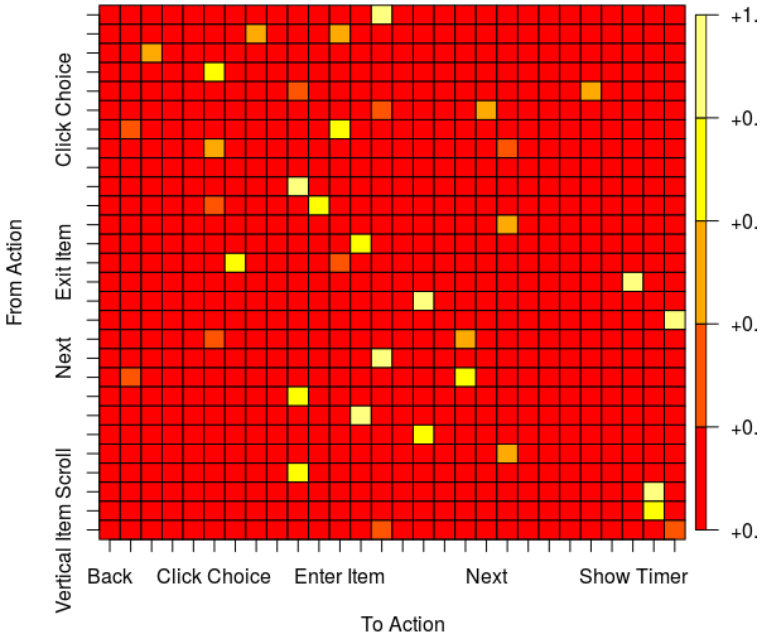


Figure 12: Markov Transition Matrix of the efficient students for question ID VH098759.

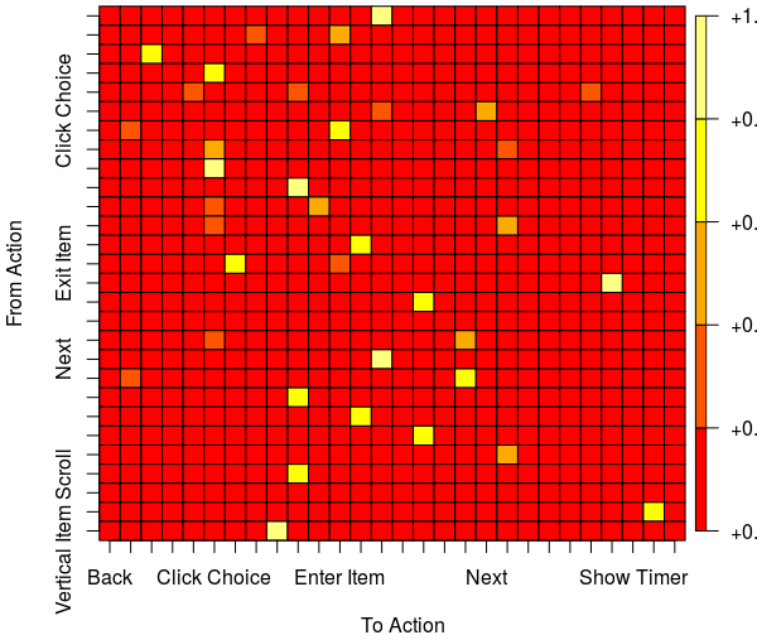


Figure 13: Markov Transition Matrix of the inefficient students for question ID VH098759.

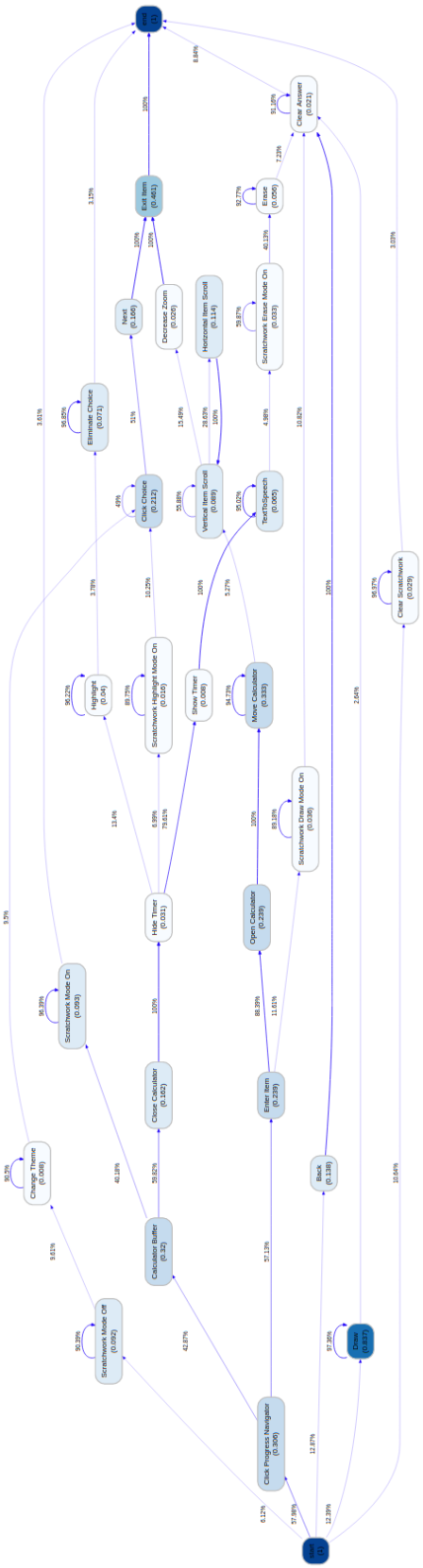

Figure 14: Process map of the efficient students for the question ID VH098759. Nodes represent possible student actions and the edges represent students doing the actions. The left and right nodes are start and end of the student interaction.

We can see that some action transitions on the far right and bottom sides of the matrix are done differently in the inefficient behavior group. Overall, the problem-level process features were not helpful for us. Our model started overfitting when we included them (we got high cross-validation accuracy on the training set, but low public validation set accuracy). It is likely, though, that if the sample size was bigger, the results would be different.

# 5. FEATURE ENGINEERING

Behavior detection problems are often formulated as classification problems, where the classes represent different types of behaviors. We took a feature based approach to build our behavior detection model. When building a classification model, we need to identify features that provide the model with more discriminatory power. In other words, a feature is good for the classification model if certain ranges of its values strongly predict certain classes. If a certain range of values for a feature correlate with multiple classes, the feature becomes less helpful and more ambiguous for the model. The feature discovery process has no set methodology, except for being guided by well-thought out hypotheses and research questions.

Based on our research questions, we created different feature sets from the data. Each set of features encoded a different type of information that aimed at addressing the research question. Since the testing data was given in 10, 20, and 30-minute subsets, we created subsets out of the training set having only the first 10, 20, and 30 minutes of the data. Some of the features were not possible to calculate for 10 and 20 minute subsets of the data. When that was the case, we omitted those features from the feature set of the specific subset. This feature sets approach allowed us to work in a team of researchers, each working to create a different set. The sets were then combined and ran through a Genetic Algorithm-based Feature Selection procedure. We first describe our Feature Sets and then the results of the Feature Selection procedure.

## 5.1. FEATURE SETS

Our features aimed to capture different aspects of the student behavior that would help us accurately predict the effective or ineffective test-taking behavior of students. For each of our research questions, we tried to come up with meaningful numeric vectors that can get at an answer to the research question. We used complex data transformation techniques available in the `tidyverse` package in the `R` environment to create the features (see Section 8 to access our code).

### 5.1.1. Time spent on each problem

Time spent on each problem was a feature set that was simple yet effective. In this feature set, we calculated the time students spent on each of the problems. Given the format of the data, this was a tricky feature to calculate. We could not simply use minimum and maximum timestamps for each of the problems, as the student could enter and exit the problem at any time. Instead, we looked at the Enter Item and Exit Item actions for each problem, and found the individual 'item session times,' which we then added to calculate the total time students spent on each item.

### 5.1.2. Percentiles of the time spent on each problem

This feature set was developed to add a compressed version of the time spent information into the model. First, we calculated the time spent by every student on every item, and then we

converted the times into percentile by item. Then, for each student, we calculated the minimum, maximum, first quartile, and third quartile of all of the percentile values of the student. The outcome generation process utilized percentiles to generate the outcomes, so we hypothesized that using percentiles to encode the student behavior was likely to correlate with the outcomes.

#### 5.1.3. Hiatus between student actions

While taking the test, we hypothesized that some students would be interacting with the digital system quickly and some would be interacting slowly, meaning that the time gaps between actions would be different for different students. Looking at the hiatus variable across all students, we saw that the times between different actions had an interquartile range from 0.16 seconds to 2.6 seconds. It was likely that the more important actions of entering the answer had a bigger hiatus related to them. We wanted to encode the distributions of students' response patterns as features, so we converted all of the hiatuses of the students into discrete buckets based on the hiatus value. The buckets we used were 0 to 0.5 seconds, 0.5 to 1 seconds, 1 to 5 seconds, 5 to 10 seconds, 10 to 20 seconds, 20 to 50 seconds, 50 to 100 seconds, and 100 seconds or more. Each bucket became a feature whose count was the number of actions for a given student in a given bucket.

#### 5.1.4. Time to select the choice in the MCQ problems

This was a feature that intended to encode how 'hasty' the student was. If the student was very quick in answering the question as soon as she saw it, we hypothesized that it would tell us something important about the test-taking behavior of the student. Since we were using tree based algorithms, it was possible that an interaction of this feature with some other feature would lead to meaningful information that the predictive model could use. We first selected the MCQ data with the 'Click Choice' action, and then looked at the time of the first 'Click Choice' within each of the MCQ problems. Then we took min, max, average, and standard deviation of the individual question-wise values.

#### 5.1.5. Student behavior similarity to effective and ineffective 'behavior prototypes'

We described earlier in Section 4.1.2 how we converted Process Models into Markov Matrices to create 'behavior prototypes' matrices. We created these prototype matrices for how students switched between the problems. First, two prototype matrices were created, one for the efficient test-takers and one for the inefficient test-takers. We then created single matrices for each of the students, and calculated the cosine similarity between individual student vectors and the prototype vectors (matrices were unrolled into vectors). We also took the difference between the similarities, because if the student vector was close to both of the prototypes, then the low difference would tell the predictive model to treat the similarity features as ambiguous.

This method of creating the features can be applied to many different process analysis scenarios. We attempted to apply this for the problem level prototype behaviors, and found the model to be overfitting. As noted earlier, the results could differ for a larger sample size. It is possible to omit the zero variance indices in the matrix when calculating the similarity, which we did not do because we did not think of it at the time. This way of encoding the process features provides us a more transparent way to build a student model.

### 5.1.6. Counts of different student actions

Counts are a very typical feature to add in most of the feature engineered approaches. Besides counts, we can use averages, minimums, maximums, and standard deviations. The main idea here is to create a large number of features, and see if any of them turn out as important during the feature selection procedure. We added counts of the following actions in this feature set:

- Open Calculator
- Click Progress Navigator
- Choice
- Scroll
- Receive Focus
- TextToSpeech
- Eliminate Choice
- Draw

### 5.1.7. Measures in different time slots of the data sample

This feature set was inspired by our Curriculum Pacing analysis. We hypothesized that if the student was showing an 'ideal' behavior, at certain points in time, she would be at a certain state as defined by our metrics. For example, if the student is patiently solving problems one by one, then in the first half of her data, the average position of her problem number would be near the problem at the 25th percentile. Similarly, other metrics would also be around some 'ideal' values at that time. To calculate the features in this set, we first divided the student data based on the time in 5 equal slots. Then, for each slot, we calculated the following features:

- Number of unique problems
- The total number of times the student entered the items
- Total calculator opens
- Total observations in the data
- Average, minimum, maximum, and standard deviation of the problem positions for the problems seen within the time bucket

Although our list of the metrics here appears limited, virtually any number of metrics that we calculated for the whole dataset can be calculated for every time slot.

### 5.1.8. Problem completion rate

This feature set consisted of just one feature, but it was a very tricky feature to calculate. No explicit information about problem completion was given to us in the data, but the metadata of the student action gave details about what the students did within the problems. Using the metadata, we could infer whether the student had completed a problem or not. For us, the completion simply meant that the student had responded to the problem. For each of the problem types, we developed our own rules to infer whether the problem was completed or not. The rules

were as follows:

- MCQ: Look for a Click Choice action
- Fill in the Blank and Multiple Fill in the Blank: Look for a Math Keypress action/
- Matching: Look for 4 distinct Drop Choice actions that had a keyword 'target' in their metadata. Matching questions required students to drag and drop items.
- Composite Constructed Response: Look into the metadata for the presence of the keyword "Part A/B/C". This was done using a Regular Expression.

The problem completion rate was the average of the completion rate of all of the problems (multi-answer problems could have a partial completion, all other problems had a binary completion value). The unseen problems that were not attempted at all were also counted as incomplete.

#### 5.1.9. Problem switching rate

While taking the test, a student can switch to work on a different problem for many reasons. Maybe she has solved the current problem, maybe she is leaving the current problem to come back later, maybe she is just reviewing the problems, or maybe she is frustrated and just switching between problems impulsively. We hypothesized that the rate of switching between the problems and the amount of going forward and backward were possibly related to the outcome variable. Specifically, we calculated the following measures for each of the students:

- Total number of forward switches (going from an earlier problem to a later problem)
- Total number of backward switches (going from a later problem to an earlier problem)
- Total problem switches
- Percentage of forward switches

#### 5.1.10. Approximate problem correctness

We tried approximating the % correct of the items by first identifying the most frequently occurring answers, and then assuming those answers were the correct answers. The process of extracting the answers from the data included Regular Expressions and data transformation tricks. However, this feature did not get selected in the feature selection procedure in the end, which was surprising to us, as earlier studies have found behavior and outcomes to be correlated. We later discovered that it was possible for us to actually see the questions based on their question ID on the NAEP website. Doing so would have given us the actual problem correctness value rather than the approximate value. It is possible that the actual correctness of the problem correlates differently with the outcome.

#### 5.1.11. Time spent doing different types of actions

This feature set was designed to capture the student behaviour based on how much time the students spent doing different actions. We hypothesized that this feature created a 'behavior profile' of the students that could help us detect the target behavior. At the time of crafting this feature, we did not consider the fact that different actions had different frequency across various

questions. We chose to ignore the problem level information and simply calculated counts of the actions that we felt were representative of the student behavior. This feature set captured how much time students spent doing the following actions:

- Selecting the choices in MCQ problems
- Using the calculator
- Looking at the timer
- Doing scratchwork, etc.

For some of these actions, outlier values can indicate the presence of certain types of engagements. For example, if a student is spending too much time changing MCQ options, maybe they are frustrated or disengaged.

#### 5.1.12. Balance between the time spent in solving earlier and later problems

This feature was developed to observe if students were taking more time at the start of the test to solve earlier problems and left themselves less time for other problems at the end of the test. This kind of student behaviour could be termed as inefficient, as ideally students should be allocating appropriate time for each problem.

To derive this feature, we calculated time taken by a student to solve each problem and then for each problem we calculated the average time taken to solve it. Then for each problem we took the ratio (time deviation ratio) of time taken by the student to the average time taken to solve the problem, where a value greater than 1 means the student took more time than average to solve the problem. Then, for each student, we arranged the problems in a temporal sequence and looked at the slope of the time deviation ratio, where a negative number would suggest that the student took more time than required at the start of the test and less time than required as they came towards the end of the test.

#### 5.1.13. Miscellaneous features

We added some more count and average types of features into our list of features. A typical rationale behind creating such features is that if you generate a large enough volume of such features, it is likely that some of them are important predictors of the outcome. In predictive modeling practice, it is not uncommon to find a good predictor that we cannot easily make sense of. For this reason, it is not a bad practice to 'throw all of the data into the model and see what happens'. This feature set was small for us, since we had derived similar features as in other feature sets. Here are the features in this set:

- Average number of actions performed per problem by students
- Average number of attempts per problem by students
- Percentage of time taken to solve the problems
- Unique number of actions done by the student

## 5.2. Data Preprocessing

At the end of the feature engineering process, we applied a set of preprocessing algorithms to the features. We found that preprocessing gave us better results on the public validation set. We applied the following list of preprocessing algorithms from the `caret` package (Kuhn and Johnson, 2013a) using its `preProc()` function:

1. Winsorization
2. Remove zero variance variables
3. Remove near zero variance variables
4. Centering
5. Scaling
6. Removing correlated variables (using the default parameters that the function provided)
7. The Yeo Johnson transform

At the end of the preprocessing, the 10, 20, and 30 minute subsets of the training data had the following number of features:

| Training Data Subset | # of Features |
|---|---|
| 10 min | 63 |
| 20 min | 73 |
| 30 min | 85 |

## 5.3. Feature Selection

Once we had our final list of features, we put them through the Genetic Algorithm (GA) based Feature Selection procedure (Kuhn and Johnson, 2013b). This was also done using the functions that the `caret` package provides. Our procedure used the Bagged Tree model for optimization, and Kappa was the target metric to optimize. We chose to find the optimal subset of features that gave us better Kappa because it would lead to a more balanced model, and Kappa was also one of the evaluation metrics for the competition. The exact details of our GA configuration are available in our code repository[8] described in Section 8.

We initially did some experiments on what number of iterations and population size to use in the GA algorithm to see what kind of solutions we got. We started by using a smaller population size (50) and a larger number of iterations (100). By trial and error, we discovered that we needed a bigger population size, but not too big. We also experimented randomly with the Crossover Probability, Mutation Probability, and Elitism parameters of the algorithm to see if they had any impact on the final outcome. Each of the GA iterations took anywhere from 10 to 18 hours to complete on a cloud-based 64-thread virtual machine. It is possible to get the same speed with an 8 or 12-thread desktop processor, if you use an appropriate configuration. Figure 15 shows the number of iterations and population sizes for each run of the GA algorithm, and also shows where we found the best solution. The best GA solution was identified by using the

[8] https://github.com/nirmalpatel/naep-competition-submission/blob/main/model/build_ga_models.R

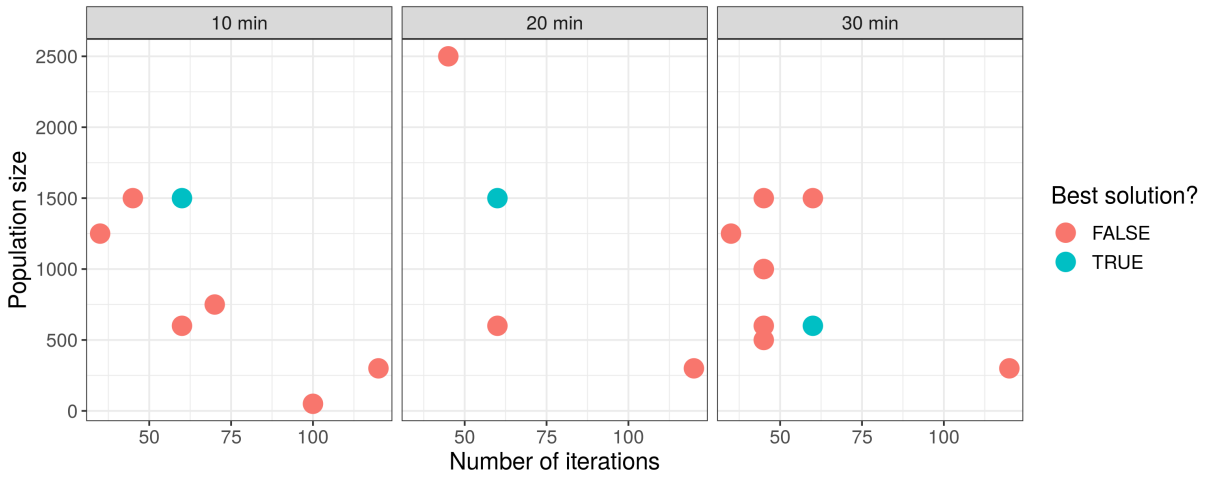


Figure 15: Results of our GA algorithm experiments.

features selected by the GA algorithm and seeing if it improved our existing public validation set results.

At the end, each subset of the features (10, 20, and 30 minutes) had the following numbers of features from the highest performing GA Feature Selection procedures:

| Training Data Subset | # of Features Selected Out of the Total |
|---|---|
| 10 min | 26 out of 63 |
| 20 min | 20 out of 73 |
| 30 min | 21 out of 85 |

Each of these subsets had features from different feature sets. Table 2 shows which feature sets had a presence in which best-performing feature subset:

From Table 2, we can see that when we have less data for prediction, we need more information about the student to make a correct prediction. When we have more data, it is likely that a fewer number of features is sufficient. However, this is just an observation in our analysis, and a broader analysis would be required to present a more generalized finding. This table also provides us insight into our research questions. We can see that two feature sets - namely, the approximate problem correctness and the balance between earlier and later problems - had none of their features selected. This means that our hypothesis about these factors being important was strongly rejected. We can see that features from some feature sets were partially selected while some feature sets were present in all three models. In a nutshell, this table provides us a proxy to evaluate our research questions.

## 6. Modeling Approach

We created separate models for each of the 10, 20, and 30 minute subsets of the training data, and used these models on the appropriate 10, 20, and 30 minute sets of the test data. This meant that we had 3 predictive models in total.

Given the smaller sample size, rather than building one complex model for each of the subsets, we decided to build an ensemble of the simpler algorithms for each of the subsets. It is

Table 2: Which feature set was present in which training data subset. X marks the presence and a blank marks the absence.

| **Feature Set** | **Selected by the GA algorithm?** | | |
|---|---|---|---|
| | **10 min** | **20 min** | **30 min** |
| Time spent on each problem | x | x | x |
| Percentiles of the time spent on each problem | x | x | |
| Hiatus between student actions | x | x | x |
| Time to select the choice in the MCQ problems | x | x | x |
| Student behavior similarity to effective and ineffective 'behavior prototypes' | x | x | x |
| Counts of different student actions | x | x | x |
| Measures in different time slots of the data sample | x | x | x |
| Problem completion rate | x | | |
| Problem switching rate | x | | x |
| Approximate problem correctness | | | |
| Time spent doing different types of actions | x | x | x |
| Balance between earlier and later problems | | | |
| Miscellaneous Features | x | | |

possible that one model is better at detecting the positive class, and another is better at detecting the negative class. In such a case, an ensemble would be an idea tool to obtain better overall accuracy. This was true for our setup, and when we started combining the predictions from different algorithms, we saw better overall results. We kept adding more models to our ensemble until we reached a point where our validation set accuracy started decreasing from having more models. Using R package `caretEnsemble` made it very easy for us to implement our ensemble model. Here are the algorithms along with their `caret` codenames[9] that we used in the ensemble:

1. Stochastic Gradient Boosting (gbm)
2. Regularized Random Forest (rrf)
3. Random Forest (rf)
4. Regularized Logistic Regression (regLogistic)
5. Distance Weighted Discrimination with Polynoial Kernel (dwdPoly)
6. k-Nearest Neighbors (knn)
7. Naive Bayes (nb)
8. Partial Least Squares (pls)
9. Support Vector Machines with Radial Basis Function Kernel (svmRadial)
10. Linear Distance Weighted Discrimination (dwdLinear)
11. Neural Network (single layer)

[9] https://topepo.github.io/caret/train-models-by-tag.html

12. Bayesian General Linear Model (bayesglm)
13. Quadratic Discriminant Analysis (qda)

The correlation of the training predictions between these algorithms and with the ensemble are given in Figure 16. We can see that the two Random Forest algorithms (RRF and rf) correlated the least with other algorithms, while they correlated perfectly with each other. We can also see the Bayesian General Linear Model algorithm (bayesglm) correlated perfectly with the Regularized Logistic Regression algorithm (regLogistic). The Bayesian General Linear Model algorithm was also used as the meta learner. Details of the parameter grid for each of the algorithms is given in our open source code[10] described in Section 8.

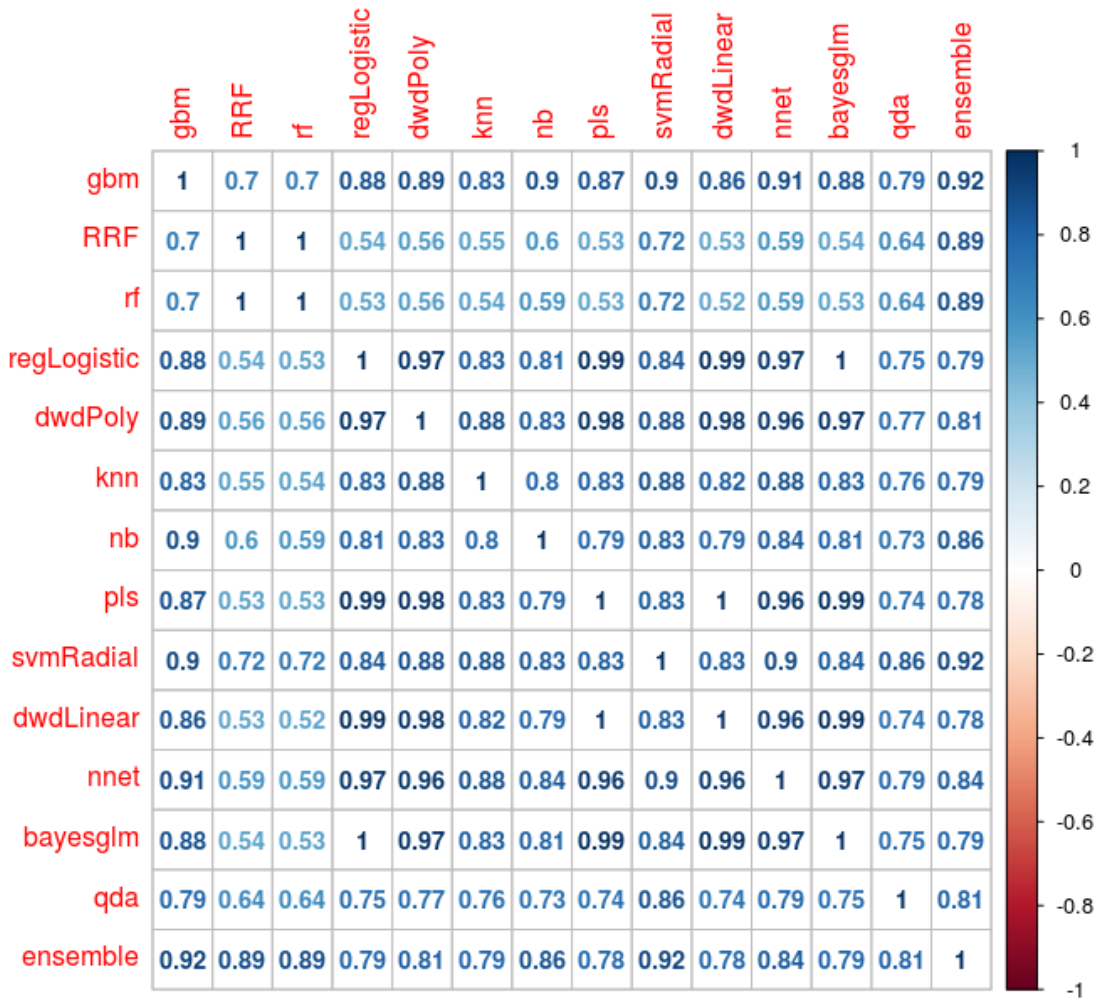


Figure 16: Correlations between the ensemble algorithms.

All of our modeling algorithms including the meta learner algorithm used a 10 fold cross-validation with 3 repeats to find the optimal model parameters. This meant that for every algorithm and parameter combination, 30 models were trained and the outcome metrics of those 30

[10] https://github.com/nirmalpatel/naep-competition-submission/blob/main/model/build_ensemble.R

different models were averaged and considered as the outcome for the given algorithm and its parameters. We decided to optimize and look for the algorithm parameters that led to the highest Kappa metric of the model.

## 7. RESULTS

We had several algorithms in our ensemble that made the predictions. Out of them all, the Stochastic Gradient Boosting (GBM) and Support Vector Machines with Radial Basis Function Kernel algorithms correlated with the ensemble predictions the most ($r = 0.92$). Out of these two algorithms, the GBM algorithm provided a measure for relative importance of the predictor variables. Figures 17, 18, and 19 show the relative importance of the predictor variables for the 10, 20, and 30-minute models.

In the figures, we can see that the 20-minute GBM model relied heavily on a smaller number of features (high importance for a few variables), the 10 minute model relied on a moderate number of features (a good distribution of the importance), and the 30-minute model had many features with high importance. Note that the variables for the modeling were selected by the Genetic Algorithm beforehand, and the GBM algorithm did not get to see the variables that were thrown out by the GA Feature Selection algorithm. One clear distinction between the 30-minute model and the other two models is that the 30-minute model had several high importance variables involving the time spent on different questions. It is likely that at the end of the 30 minutes, the distribution of the time spent on the questions is highly indicative of the student behavior in the time period that is adjacent (Block B). For the 10-minute model, since the student has not fully allocated the times to each of the question, and we don't know how they will go about doing it, features other than the time spent features remain more important. In the 20-minute model, it is at once both surprising and interesting to see that the 'behavior prototype' related features have a very high importance. Given that the variable selection was based on a stochastic algorithm, we cannot be sure that this was the globally optimal set of variables.

### 7.1. COMPETITION OUTCOME

In the final evaluation, our winning model[11] stood at #2 on both the public validation set and the private test set. This correspondence between these formative and summative scoreboards was notable: the #18 model on the public validation set scored #1 on the final score board. The top-scoring model on the public validation set dropped to #3 on the final score-board. These evaluations were done on two metrics: Adjusted AUC and Adjusted Kappa. They were defined by the competition organizers as follows:

$$AdjustedAUC = \begin{cases} 0 & ,if\ AUC < 0.5 \\ 2(AUC - 0.5) & ,otherwise \end{cases}$$

$$AdjustedKappa = \begin{cases} 0 & ,if\ kappa < 0 \\ kappa & ,otherwise \end{cases}$$

The final ranking was done by a sum of Adjusted AUC and Adjusted Kappa. The outcome metrics for our solution are given in the Table 3. From the Adjusted AUC formula, we can

[11] https://sites.google.com/view/dataminingcompetition2019/winners

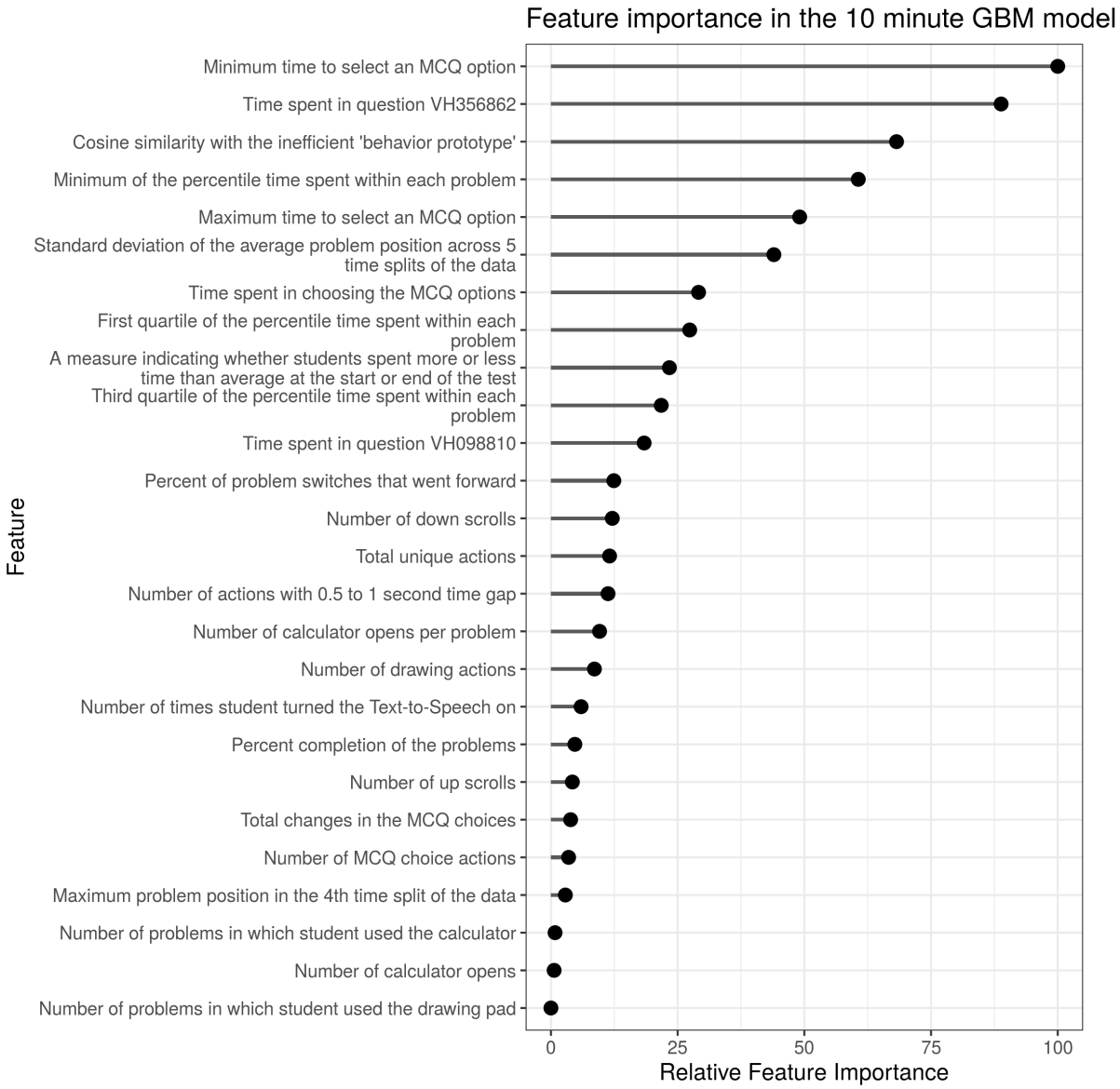


Figure 17: Relative variable importance for the 10 minute model variables.

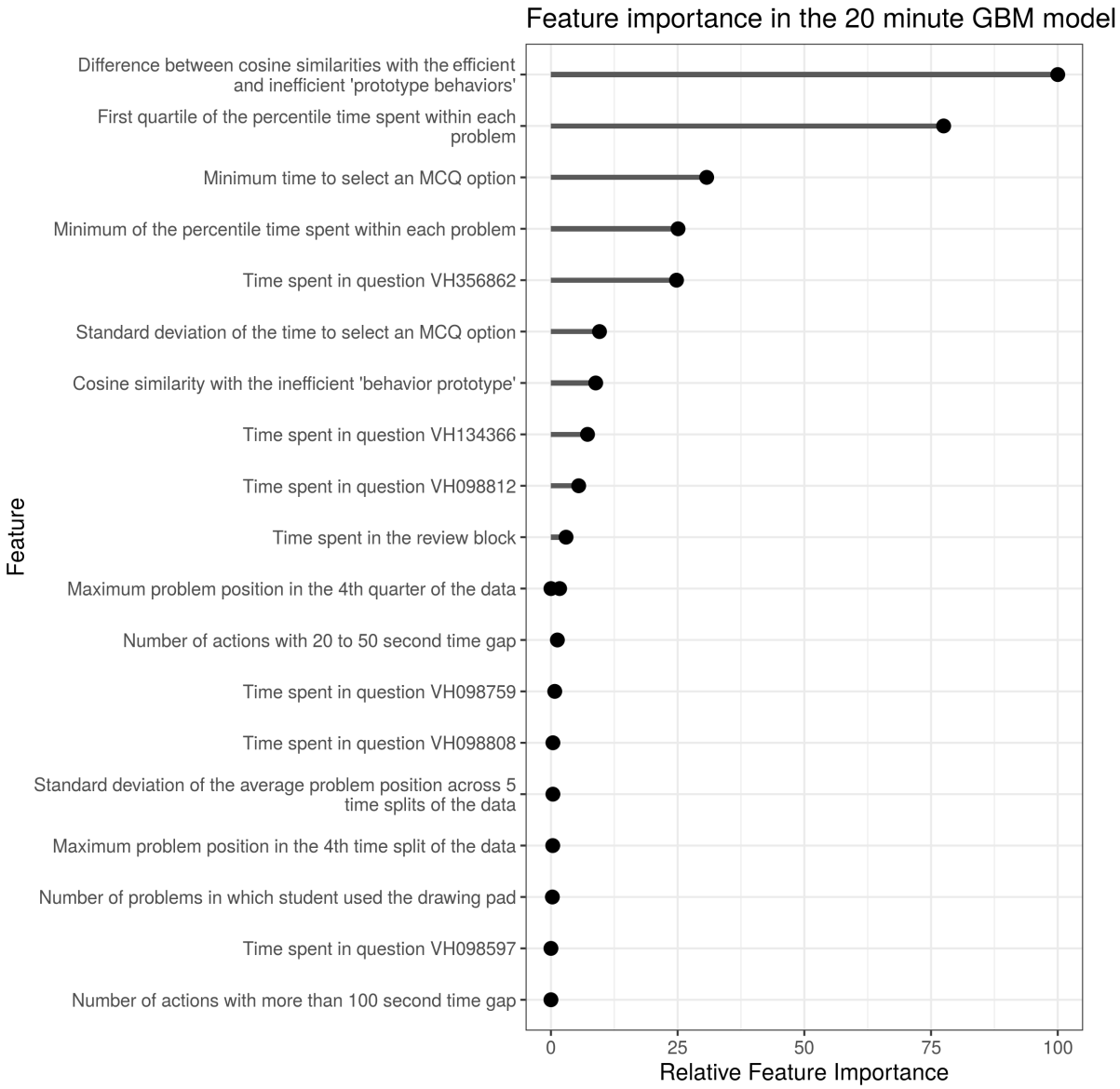


Figure 18: Relative variable importance for the 20 minute model variables.

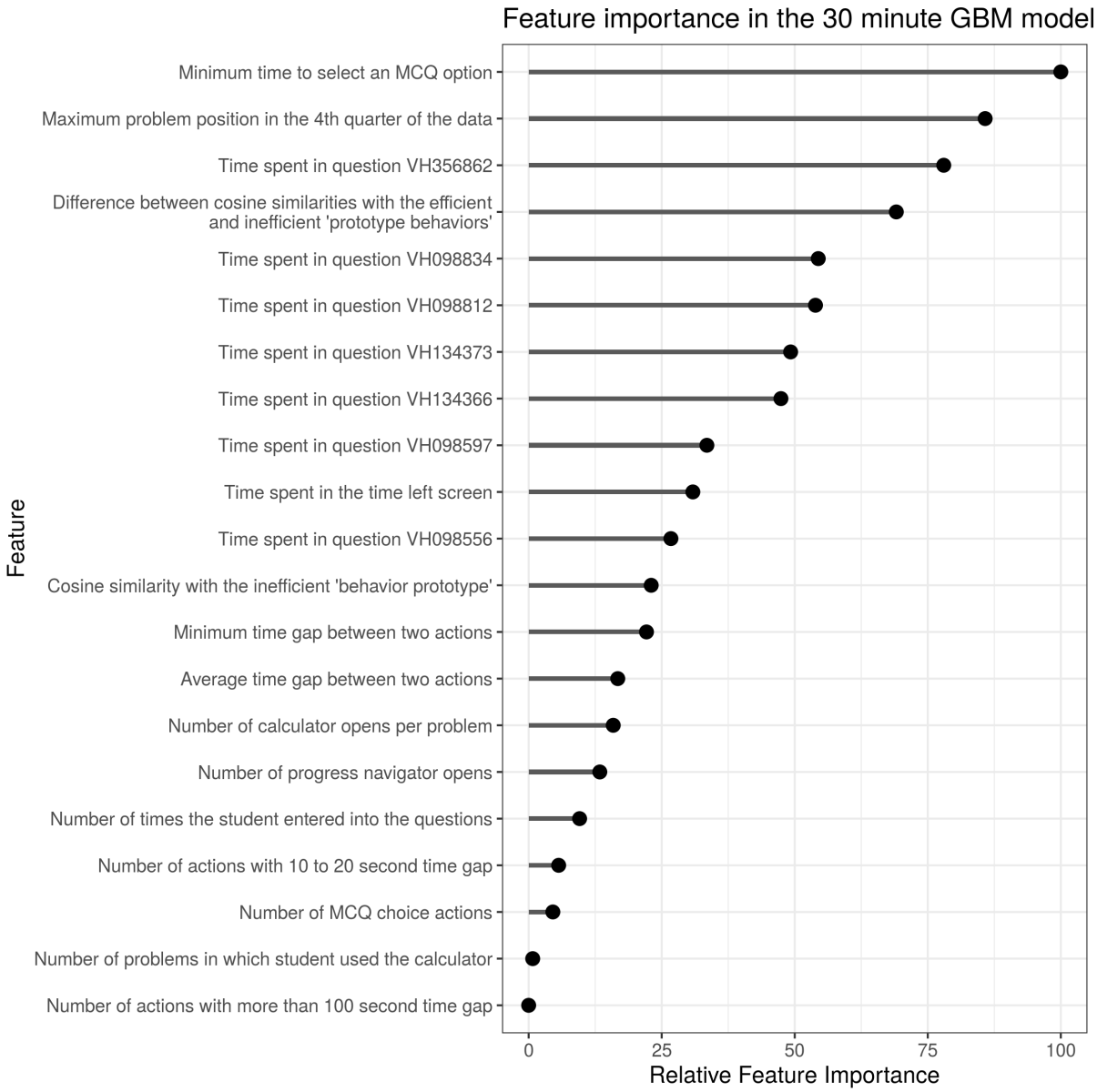


Figure 19: Relative variable importance for the 30 minute model variables.

Table 3: Results of the competition for our solution.

| Set | Adjusted AUC | Adjusted Kappa | Aggregated Score |
|---|---|---|---|
| Public Validation Set | 0.3660 | 0.2617 | 0.6277 |
| Private Test Set* | 0.3257 | 0.2268 | 0.5524 |

* Used for final ranking

see that our model's actual AUC was 0.66. The AUC can be interpreted as the probability that a randomly chosen positive example (efficient behavior) is ranked above a randomly chosen negative example (inefficient behavior), according to the model's internal value for the examples. In the predictive modeling context, the Kappa value can be interpreted as how well the classifier performed, controlling for the baseline accuracy. One specific interpretation of the Kappa metric would tell us that on the test set, our solution achieved a rate of classification 22.7% of the way between whatever the baseline accuracy was and 100% accuracy. On the training set, the baseline accuracy (or the proportion of the majority class, which was the efficient behavior) was 60%.

## 8. Open-Source Code

The open-source R code that can fully reproduce our features, models, and the final outcomes is available at https://github.com/nirmalpatel/naep-competition-submission. The code does not contain the raw NAEP process data, which is the only addition required to run our code. If the reader has access to a different or a larger sample of NAEP process data in the same format as described in Table 1, our code will work directly with it. Please note that the model building process is time-consuming, and on a 4 GHz 8 Thread Intel CPU, it generally takes around 1.5 hours to train the ensemble.

## 9. Discussion

Some of the more interesting features of our solution were the similarities of individual students with the efficient and inefficient 'behavior prototypes'. Here, we had a very specific set of behaviors, taken from a specific set of students, that nevertheless appeared to generalize to the wider set of students. To examine other aspects of these features, we looked at how they correlated with the full feature set. The correlations changed considerably from the 10-minute dataset to the 30-minute dataset, so we restricted our correlation analysis to the 30 minute dataset because by that time, all of the student activity in the block was completed.

The first thing we noticed was that the correlation of student similarity to the efficient and inefficient 'behavior prototypes' was almost always the same, meaning that if a feature was positively correlated with efficient behavior similarity, then it was also as positively correlated with inefficient behavior similarity. However, the difference of the similarities with the efficient and inefficient 'behavior prototypes' almost never correlated in the same way as the individual similarities. We also noted from the feature selection that the difference was much more important than the individual similarities in the 20 and 30-minute models, while the 10-minute model had the inefficient behavior similarity as more important. It is very likely that the process data of a single student has both efficient and inefficient processes within it, and a big absolute

value of the difference feature will tell us how 'one sided' the student behavior was. In other words, larger distance between the similarities means that the student was either more efficient or more inefficient. In our case, we calculated the difference as the Efficient Similarity - Inefficient Similarity. This can be seen as the 'adjusted similarity to the efficient behavior.' Here are two interesting correlations that the adjusted similarity to the efficient behavior had with other features:

- We noticed that the adjusted similarity to the efficient behavior was significantly negatively correlated ($R = -0.63$) with the progress navigator opens, probably telling us that the efficient students used the progress navigator less.
- We also saw that the adjusted efficient behavior similarity was significantly correlated ($R = 0.62$) with the percentage of forward problem switches. This may tell us that efficient students went forward more.

Although we can learn about such differences using the provided outcomes, seeing them via the novel 'behavior prototype' calculation method tells us that computing the similarity of an individual student to different behavioral patterns is a good way to generate important behavior detection features.

We also looked closely at the Minimum Time to Select an MCQ Option (Minimum TTC or Minimum Time to Choice) feature which was the most important feature for both the 10-minute and 30-minute models. It turned out that a number of features representing how much problem switching and navigating students did were negatively correlated with the Minimum TTC ($R$ ranging from -0.30 to -0.42). This probably indicates that students who frequently switched between items during the test were more likely to rapidly click an MCQ option (or vice versa). It is possible, though, that efficient students also quickly switch through many problems, just because they have left out some problems to finish at the end. In this case, their Minimum TTC would be high, even with a high problem switching rate. In fact, a two sample t-test showed that the efficient students as marked by the training labels had a significantly higher Minimum TTC than the inefficient students (Average Efficient Min TTC = 12.7 seconds, Average Inefficient Min TTC = 10.7 seconds, $t = 5.2$, $p = 0.0$, calculated on Winsorized samples).

One of the things that we found puzzling in the Curriculum Pacing analysis was the frequent up-down switches between adjacent problems. Given that many students showed a straight path from start to finish, the up-down switches should not be a feature of bad data. In that case, such up-down pattern might indicate that the student is trying to decide which problem to work on, impatiently or otherwise. For example, we can see that Student 31 in Figure 7 seems to flip around twice across a two set of problems. In the same plot we can also see that many students exhibit such flipping around behavior through the test. If such question flipping behavior exists at scale, we need to understanding it from the students about why they do it. Is it to decide which question they should work on? If they are equally unsure about which question they need to work on, maybe they can randomly pick one rather and give it a try. And if such random picking strategy turns out to be a good one for the sake of test-taking, additional UI improvements can be made into the NAEP Digital Test where frequent flipping between the questions can lead up to a prompt telling the students that if they are unsure what they need to work on, they can just pick one question and focus on it for a while.

We hypothesize that the features representing the time spent on different problems can generalize across multiple behavior detection scenarios. Our hypothesis is based on 3 different

observations. First: We saw that the #1 winning entry of the competition only had time spent on problem features, and while it was at a distant rank of 18th in the validation set, the same entry came up at the top of the test set. Second: We also started our model building by using the time spent on the problem features. These features gave us an aggregated score of about 0.53 on the validation set; by contrast, our final model gave us an aggregated score of about 0.63, showing us that about 80% of the 'modeling work' was done by just these features. Third: A study by (Şahin, 2019) also showed that using the time spent on problem was useful in predicting student outcomes of the NAEP test. These observations tell us that the time spent on problem features might be more generalizable to different behavior detection scenarios. We also hypothesize that there might be a way to improve the generalization of the time spent on problem features by normalizing the time, as some problems will generally take more or less time to complete than the others.

## 10. Conclusion and Future Work

Overall, this work provided us with positive feedback about the future of Process Analysis in the field of Educational Data. As Learning Analytics practitioners, we knew about the importance of Process Analysis, but during this challenge, we discovered that not only can Process Analysis tell us meaningful data stories (which is of utmost importance in for educational data practice), it can also help us build high-performance predictive models. This leads us to believe that Process Analysis is a promising paradigm for exploring student data and generating novel hypotheses about student learning and behavior.

Our feature selection routine left out many features that we derived during our initial data analysis process. Doing a factor analysis of the full feature set might reveal different latent factors that are present in the dataset. A more complex study to understand these latent factors may require the use of techniques like Structural Equation Modeling. A preliminary factor analysis on the 30-minute model features revealed one general factor representing what appeared to be the overall level of student engagement with the test and its items. This underlying factor had various interesting features loaded on it, such as hiatus distribution, 'behavior prototype' related features, and how much time students spent interacting with the MCQ options. This preliminary finding was interesting and may lead to further studies of test-taking engagement patterns.

In our Curriculum Pacing analysis, we were able to observe several patterns in the variety of test-taking students. Some students moved through the test in a linear fashion, some jumped around, and some even worked backwards from the end. The reader might be able to personally relate to the experiences of test-takers when observing the patterns presented in Figures 6 and 7. It would be very interesting to run a clustering analysis on a larger sample of the NAEP data to discover the test-taking processes exhibited by the students at scale, and see if any of the clusters are related with high or low outcomes. Surely, the patterns are not the causes but only the symptoms of the different levels of student knowledge and cognitive function. And, of course, it would be extremely useful to run similar analyses that involve the outcomes of student performance on individual items. For various reasons related to privacy and test validity, this competition could not disclose student demographics, item performance data or details about the items (wording or options). We strongly believe that Process Analysis of a larger and more comprehensive sample of the NAEP data can lead to many interesting Learning Science discoveries. For instance, we anticipate the utility of using process analysis to understand the

effects of student disabilities and NAEP testing accommodations. We also expect that Process Analysis methods could inform test-taking UI improvements which could support more efficient test-taking behaviors by all students.

## 11. ACKNOWLEDGMENTS

Many thanks to the competition organizers for motivating this research in the Educational Data Mining community. We hope that our work on Educational Process Analysis can support and inspire others.

## REFERENCES

ALEVEN, V. AND KOEDINGER, K. R. 2000. Limitations of student control: Do students know when they need help? In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, G. Gauthier, C. Frasson, and K. VanLehn, Eds. Springer, Berlin, Heidelberg, 292–303.

ALEVEN, V., MCLAREN, B., ROLL, I., AND KOEDINGER, K. 2006. Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education 16,* 2, 101–128.

BAKER, R., WOOLF, B., KATZ, I., FORSYTH, C., AND OCUMPAUGH, J. 2019. Nation's Report Card Data Mining Competition 2019. https://sites.google.com/view/dataminingcompetition2019/home.

BAKER, R. S. 2015. *Behavior Detection, Big Data and Education*, 2nd ed. Teachers College, Columbia University, New York, NY, Chapter 3.

BAKER, R. S., CORBETT, A. T., AND KOEDINGER, K. R. 2004. Detecting student misuse of intelligent tutoring systems. In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems*, J. C. Lester, R. M. Vicari, and F. Paraguaçu, Eds. Lecture Notes in Computer Science, vol. 3220. Springer, 531–540.

BAKER, R. S., CORBETT, A. T., KOEDINGER, K. R., EVENSON, S., ROLL, I., WAGNER, A. Z., NAIM, M., RASPAT, J., BAKER, D. J., AND BECK, J. E. 2006. Adapting to when students game an intelligent tutoring system. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, M. Ikeda, K. D. Ashley, and T.-W. Chan, Eds. Springer, 392–401.

BAKER, R. S., CORBETT, A. T., KOEDINGER, K. R., AND WAGNER, A. Z. 2004. Off-task behavior in the Cognitive Tutor classroom: When students "game the system". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, E. Dykstra-Erickson and M. Tscheligi, Eds. ACM, 383–390.

BAKER, R. S. J. D. 2007. Modeling and understanding students' off-task behavior in intelligent tutoring systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, M. B. Rosson and D. J. Gilmore, Eds. ACM, 1059–1068.

BERGNER, Y. AND VON DAVIER, A. A. 2019. Process data in NAEP: Past, present, and future. *Journal of Educational and Behavioral Statistics 44,* 6, 706–732.

BOGARÍN, A., CEREZO, R., AND ROMERO, C. 2018. A survey on educational process mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8,* 1, e1230.

BOGARÍN, A., ROMERO, C., CEREZO, R., AND SÁNCHEZ-SANTILLÁN, M. 2014. Clustering for improving educational process mining. In *Proceedings of the 4th Learning Analytics and Knowledge*

*Conference*, M. D. Pistilli, J. Willis, D. Koch, K. E. Arnold, S. D. Teasley, and A. Pardo, Eds. ACM, 11–15.

CETINTAS, S., SI, L., XIN, Y. P. P., AND HORD, C. 2009. Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques. *IEEE Transactions on Learning Technologies 3,* 3, 228–236.

DATA QUALITY CAMPAIGN. 2018. What parents and teachers think about educational data. https://dataqualitycampaign.org/resource/what-parents-and-teachers-think-abouteducation-data/.

GARCÍA, E., ROMERO, C., VENTURA, S., DE CASTRO, C., AND CALDERS, T. 2010. *Association rule mining in learning management systems*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, Boca Raton, FL, Chapter 7, 93–106.

GOLDHAMMER, F., MARTENS, T., CHRISTOPH, G., AND LÜDTKE, O. 2016. Test-taking engagement in PIAAC. Tech. Rep. 133, Organisation for Economic Co-operation and Development, Paris.

GÜNTHER, C. W. AND VAN DER AALST, W. M. P. 2007. Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International Conference on Business Process Management*. Springer, 328–343.

GUO, H., RIOS, J. A., HABERMAN, S., LIU, O. L., WANG, J., AND PAEK, I. 2016. A new procedure for detection of students' rapid guessing responses using response time. *Applied Measurement in Education 29,* 3, 173–183.

HOLSTEIN, K., MCLAREN, B. M., AND ALEVEN, V. 2018. Student learning benefits of a mixed-reality teacher awareness tool in AI-enhanced classrooms. In *Proceedings of the 19th International Conference on Artificial Intelligence in Education*, C. P. Rosé, R. M. Maldonado, H. U. Hoppe, R. Luckin, M. Mavrikis, K. Porayska-Pomsta, B. M. McLaren, and B. du Boulay, Eds. Lecture Notes in Computer Science, vol. 10947. Springer, 154–168.

KONG, X. J., WISE, S. L., AND BHOLA, D. S. 2007. Setting the response time threshold parameter to differentiate solution behavior from rapid-guessing behavior. *Educational and Psychological Measurement 67,* 4, 606–619.

KUHN, M. AND JOHNSON, K. 2013a. *Applied Predictive Modeling*. Springer, New York, NY.

KUHN, M. AND JOHNSON, K. 2013b. An introduction to feature selection. In *Applied Predictive Modeling*. Springer, New York, NY, 487–519.

LEE, Y.-H. AND JIA, Y. 2014. Using response time to investigate students' test-taking behaviors in a NAEP computer-based study. *Large-scale Assessments in Education 2,* 1, 1–24.

LYNCH, C. F., BARNES, T., XUE, L., AND GITINABARD, N. 2017. Graph-based educational data mining. In *Proceedings of the 10th International Conference on Educational Data Mining*, X. Hu, T. Barnes, A. Hershkovitz, and L. Paquette, Eds. International Educational Data Mining Society.

PATEL, N., SELLMAN, C., AND LOMAS, D. 2017. Mining frequent learning pathways from a large educational dataset. *arXiv preprint arXiv:1705.11125*.

PATEL, N., SHARMA, A., SELLMAN, C., AND LOMAS, D. 2018. Curriculum pacing: A new approach to discover instructional practices in classrooms. In *Proceedings of the 14th International Conference on Intelligent Tutoring Systems*, R. Nkambou, R. Azevedo, and J. Vassileva, Eds. Lecture Notes in Computer Science, vol. 10858. Springer, 345–351.

PATIKORN, M. AND HEFFERNAN, N. 2019. Dataset. https://sites.google.com/view/dataminingcompetition2019/dataset#h.p_ndPGLzSaEKFb.

PEDRO, M. O. C. Z. S., BAKER, R. S. J. D., AND RODRIGO, M. M. T. 2011. Detecting carelessness through contextual estimation of slip probabilities among students using an intelligent tutor for mathematics. In *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, G. Biswas, S. Bull, J. Kay, and A. Mitrovic, Eds. Lecture Notes in Computer Science, vol. 6738. Springer, 304–311.

PEDRO, M. O. C. Z. S., RODRIGO, M. M. T., AND BAKER, R. S. J. D. 2011. The relationship between carelessness and affect in a Cognitive Tutor. In *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction*, S. K. D'Mello, A. C. Graesser, B. W. Schuller, and J. Martin, Eds. Lecture Notes in Computer Science, vol. 6974. Springer, 306–315.

ROWE, J. P., MCQUIGGAN, S. W., ROBISON, J. L., AND LESTER, J. C. 2009. Off-task behavior in narrative-centered learning environments. In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, V. Dimitrova, R. Mizoguchi, B. du Boulay, and A. C. Graesser, Eds. Frontiers in Artificial Intelligence and Applications, vol. 200. IOS Press, 99–106.

ŞAHIN, F. 2019. Exploring the relations between students' time management strategies and test performance. Paper presented at the Annual meeting of the National Council for Measurement in Education.

ŞAHIN, F. AND COLVIN, K. F. 2020. Enhancing response time thresholds with response behaviors for detecting disengaged examinees. *Large-scale Assessments in Education 8,* 5, 1–24.

SLATTOW, G. 1977. Demonstration of the PLATO IV computer-based education system. Final report. January 1, 1972-June 30, 1976. Tech. rep., University of Illinois.

TRČKA, N., PECHENIZKIY, M., AND VAN DER AALST, W. 2010. *Process mining from educational data*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, Boca Raton, FL, Chapter 9, 123–142.

WALONOSKI, J. A. AND HEFFERNAN, N. T. 2006. Detection and analysis of off-task gaming behavior in intelligent tutoring systems. In *Intelligent Tutoring Systems*, M. Ikeda, K. Ashley, and T.-W. Chan, Eds. Lecture Notes in Computer Science, vol. 4053. Springer Berlin Heidelberg, 382–391.

WIXON, M., DE, B. R. S. J., GOBERT, J. D., OCUMPAUGH, J., AND BACHMANN, M. 2012. WTF? Detecting students who are conducting inquiry without thinking fastidiously. In *Proceedings of the 20th International Conference on User Modeling, Adaptation, and Personalization*, J. Masthoff, B. Mobasher, M. C. Desmarais, and R. Nkambou, Eds. Lecture Notes in Computer Science, vol. 7379. Springer, 286–296.

ZHOU, M., XU, Y., NESBIT, J. C., AND WINNE, P. H. 2010. *Sequential pattern analysis of learning logs: Methodology and applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, Boca Raton, FL, Chapter 8, 107–121.