Automatic Interpretable Personalized Learning

Ethan Prihar
ebprihar@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

Aaron Haim
ahaim@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

Adam Sales
asales@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

Neil Heffernan
nth@wpi.edu
Worcester Polytechnic Institute
Worcester, MA, USA

ABSTRACT

Personalized learning stems from the idea that students benefit from instructional material tailored to their needs. Many online learning platforms purport to implement some form of personalized learning, often through on-demand tutoring or self-paced instruction, but to our knowledge none have a way to automatically explore for specific opportunities to personalize students' education nor a transparent way to identify the effects of personalization on specific groups of students. In this work we present the Automatic Personalized Learning Service (APLS). The APLS uses multi-armed bandit algorithms to recommend the most effective support to each student that requests assistance when completing their online work, and is currently used by ASSISTments, an online learning platform. The first empirical study of the APLS found that Beta-Bernoulli Thompson Sampling, a popular and effective multi-armed bandit algorithm, was only slightly more capable of selecting helpful support than randomly selecting from the relevant support options. Therefore, we also present Decision Tree Thompson Sampling (DTTS), a novel contextual multi-armed bandit algorithm that integrates the transparency and interpretability of decision trees into Thomson sampling. In simulation, DTTS overcame the challenges of recommending support within an online learning platform and was able to increase students' learning by as much as 10% more than the current algorithm used by the APLS. We demonstrate that DTTS is able to identify qualitative interactions that not only help determine the most effective support for students, but that also generalize well to new students, problems, and support content. The APLS using DTTS is now being deployed at scale within ASSISTments and is a promising tool for all educational learning platforms.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI**; • **Theory of computation → Online learning algorithms**.

L@S '22, June 1–3, 2022, New York City, NY, USA.

ACM ISBN 978-1-4503-9158-0/22/06.
https://doi.org/10.1145/3491140.3528267

KEYWORDS

Contextual Bandits, Multi-Armed Bandits, Personalized Learning, Intelligent Tutoring Systems

ACM Reference Format:
Ethan Prihar, Aaron Haim, Adam Sales, and Neil Heffernan. 2022. Automatic Interpretable Personalized Learning. In *Proceedings of the Ninth ACM Conference on Learning @ Scale (L@S '22), June 1–3, 2022, New York City, NY, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3491140.3528267

1 INTRODUCTION

Personalized learning revolves around providing each student with the instruction that best suits them. At the core of personalization is the idea that there exists two groups of students, Group A and Group B, and that Group A benefits more from one teaching method, Method X, than another method, Method Y, and Group B benefits more from Method Y than Method X. This relationship, referred to as a qualitative interaction, can involve different instructional mechanisms, changing the pace of instruction, and using different evaluation methods, all with the goal of helping each student achieve their full potential. This is already difficult to achieve in small settings where teachers can directly interact with students. Effectively implementing personalized learning throughout an entire online learning platform is even more difficult.

While some platforms have created infrastructure to evaluate individual research questions regarding personalized learning [13, 18], if one instructional method is better than all other methods for all groups of students, then at least half of the students in the study received sub-optimal support, which could negatively impact the students as well as the long-term adoption of the platform. Multi-armed bandit algorithms can be used to adjust how often students receive each support option by estimating each option's effectiveness and intentionally giving more students the most effective option. Simulations using real student data found that across 22 educational experiments, using multi-armed bandit algorithms to assign students to conditions statistically significantly increased students' assignment completion rates and proficiency [16].

To automate the discovery of opportunities to personalize learning and increase students' performance within online learning platforms, this work presents the **Automatic Personalized Learning Service (APLS)**. A fully deployed service in ASSISTments, an online learning platform, that provides personalized support to struggling students upon their request. The APLS works by linking support content to aspects of students' learning environment, such

as their current assignment, problem, or prior mistakes. After compiling the relevant support content, the APLS uses a multi-armed bandit algorithm to determine which content is likely to be most helpful to the student, and then provides the selected support to the student. Two examples of student supports that can be shown to a student are shown in Figure 1.

The APLS provides student support content throughout ASSISTments whenever multiple instances of support are available, enabling multi-armed bandit algorithms to personalize students' experience at scale. However, the results of the initial APLS study using Beta-Bernoulli Thompson Sampling[21] as the recommendation algorithm were not as positive as expected. Therefore, in addition to the APLS, this work presents **Decision Tree Thompson Sampling (DTTS)**, a novel contextual multi-armed bandit algorithm designed to address the specific limitations of attempting to personalize learning within online learning platforms in an interpretable way. DTTS integrates the transparency and interpretability of decision trees into the multi-armed bandit framework and was shown in simulation to be more effective than existing multi-armed bandit algorithms when used to personalize students' support within an online learning environment.

This paper seeks to accomplish the following objectives and in doing so, provide a method for researchers to integrate personalized learning into other platforms while informing educators of the insight gained through this method:

(1) Provide a description of the APLS.
(2) Report on the results of the initial APLS study.
(3) Provide a novel algorithm (DTTS) to address the difficulties encountered when attempting to personalize students' support.
(4) Simulate the effectiveness of DTTS in realistic scenarios.

2 PRIOR WORK

There is an abundance of online platforms that attempt to personalize learning either by adjusting students' lessons based on their skill level or adapting to students' needs. McGraw-Hill Thrive, Lexia, PracTutor, HMH FUSE, Carnegie Learning's Cognitive Tutor, AutoTutor, and ASSISTments all claim to implement one or both of these methods of personalized learning [5]. However, the effectiveness of most of these platforms' personalized learning methodology has not been empirically evaluated in a classroom setting, and when evaluated, either does not show consistent positive results, as was the case with HMH FUSE [5], or achieves positive results through the use of the platform as a whole, without directly evaluating the benefit of personalizing students' experience using the platform, as was the case with ASSISTments [20] and Carnegie Learning's Cognitive Tutor [14].

Although most online learning platforms have shown little evidence for the effectiveness of personalizing students' education, there are some encouraging results as well. For example, a randomized controlled experiment done in ASSISTments found that high-knowledge students learned more from being given an entire explanation of their mistakes, while low-knowledge students learned more from being given smaller instructional segments [17]. Additionally, a meta-analysis of studies measuring the learning gains of students when grouping them by ability level found that the average effect size in 21 studies in which the students were given different instructional material was more than twice the average effect size of 30 studies in which the instructional material remained the same for each group of students [10]. The ASSISTments study and the meta-analysis both support the idea that personalizing educational content based on students knowledge level increases students' learning, but do not evaluate the effect of this personalization at scale.

While most platforms have not evaluated the overall benefits of personalization, some have allowed for testing individual hypotheses regarding personalization. The MOOClet Framework and ASSISTments have already taken steps to allow for researchers to create and deploy studies of different tutoring methods within their platforms. The MOOClet Framework provides educators with the ability to create multiple sets of material for their courses and evaluates the most effective content through randomization [18]. The ASSISTments E-TRIALS TestBed allows researchers to create modified versions of problem sets. These modified problem sets include internal random assignment of supportive content, enabling randomized controlled experimentation across all the students in ASSISTments that are assigned these problem sets [13]. In addition to allowing researchers to create randomized controlled experiments, ASSISTments also crowdsources student supports from teachers that use the platform and distributes these supports through a program called TeacherASSIST. Relevant student supports are randomly selected and provided to students upon their request, effectively creating a randomized controlled experiment in situations where more than one teacher has created support for the same problem [15]. While these platforms offer ways to gather data and test individual hypotheses, they do not automatically evaluate and deploy candidate methods for personalized learning to students throughout the platform.

3 APLS ARCHITECTURE

The APLS is designed to facilitate personalized learning in a modular way, such that regardless of the available context or student support options, the system can make an intelligent and informed decision as to which support is likely to be most beneficial to students. The APLS has two components, an online and an offline component. The online component is responsible for receiving and responding to support requests. The APLS uses the content of support requests to retrieve context it has stored on the students' learning environment, identify the potential student supports it can return, and select a multi-armed bandit model to determine which support is likely to be most effective. After a model has been chosen, potential student supports have been identified, and context on the student supports and students' environment has been gathered, the APLS uses this information to predict which student support is likely to have the most positive effect on learning. After a student support is selected, the APLS sends its prediction to the ASSISTments Tutor, which displays the support to the student.

The offline component facilitates updating the multi-armed bandit models and the context used by the models during low-load periods, e.g., at night when students that use ASSISTments are asleep. The offline component first determines the effectiveness of

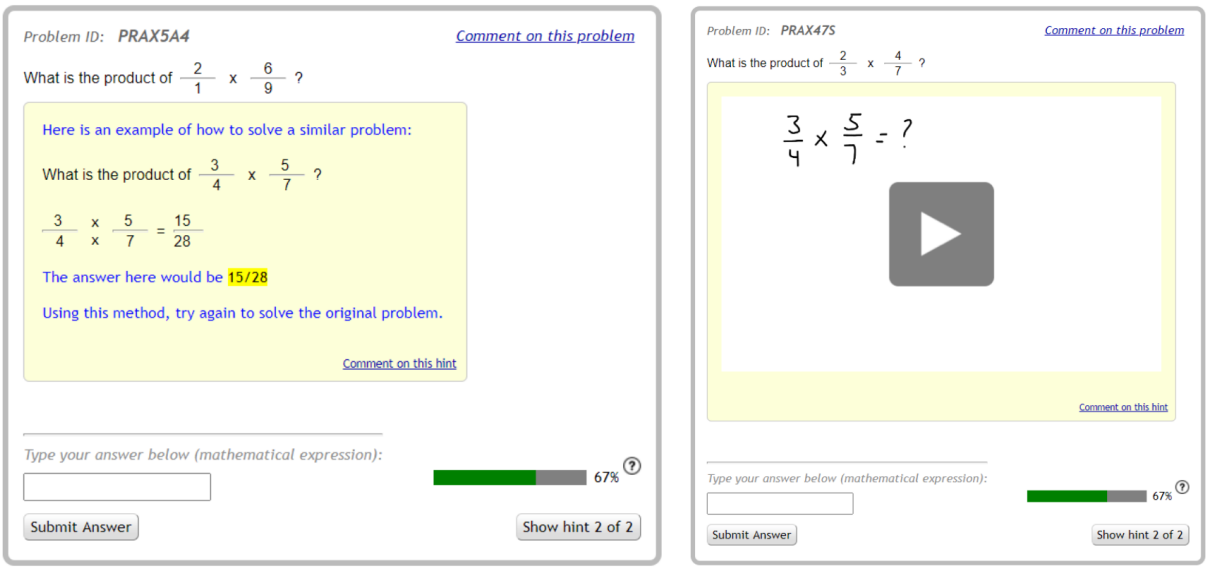

Figure 1: Two views of the ASSISTments tutor in which a student has requested support and received one of two available student supports. The student support on the left is a written explanation, and the student support on the right is a video explanation.

each recommendation made by the APLS during the day by reviewing logs of students' actions, then it uses this information to update the multi-armed bandit models. Lastly, it updates any context based on the same logs of student's actions. These offline updates allow the APLS to learn over time how to most effectively personalize students' learning. A flowchart of the online and offline tasks of the APLS is shown in figure 2.

3.1 Data Collection

In ASSISTments, as students complete their assigned problem sets, each action the student takes is recorded. This includes their use of student supports, correct and incorrect responses to problems, and duration of their engagement with various aspects of their assignments. Offline, these action logs are used to aggregate statistics on students and problems within ASSISTments, these statistics, referred to as the learning environment context, are used by the multi-armed bandit models in the APLS. The action logs are also used to determine the reward for each time a model made a recommendation, which the model uses to adjust how it makes recommendations in order to maximize reward.

Every night, the APLS collects data from the ASSISTments action logs and creates context for the recommendation models to use the following day. This context contains statistics on students' prior performance, statistics on the prior performance of all students across each problem, qualities of the problem such as text length, the skills required to solve it, its structure, e.g. multiple choice or short answer, whether it uses a diagram, and more. In addition to collecting statistics on various aspects of students' learning environments, the APLS collects information on each student support available to students. When new student supports are created, their format and HTML bodies are used to extract context such as the length of the text in the student support, whether the student support uses videos or images, whether the student support contains a question, and more. After each nightly update, the context of each student's learning environment and all student supports is updated, this ensures that the following day the models in the APLS can use the context to make recommendations. The full context of all learning environments and student supports collected by the APLS during its initial trial is hosted by the Open Science Foundation and can be found at https://osf.io/9pgv5/, and a description of all the context features used by the APLS can be found in Appendix A.

In addition to updating the learning environment and student support context, every night the reward is calculated for each recommendation made by the APLS using Algorithm 1. Algorithm 1 returns 1 when the next graded problem that the student had the opportunity to complete was completed correctly on the student's first attempt. When a student completes the next graded problem incorrectly, or when they fail to complete the next graded problem, Algorithm 1 returns 0. In cases when the student did not have an opportunity to complete a graded problem within the same assignment after requesting support, e.g., if the student requested support on the last problem in an assignment, or if all the following problems were ungraded open response problems, Algorithm 1 returns *null*. When the reward is *null*, nothing was learned about the quality of the student support, and therefore the multi-armed bandit model is not updated based on that recommendation.

3.2 Content Recommendation

In real-time, as students use the ASSISTments Tutor, they can request support. Whenever a student starts a problem, the APLS is sent a support request and uses a multi-armed bandit algorithm to return the support it predicts will be most likely to result in a high

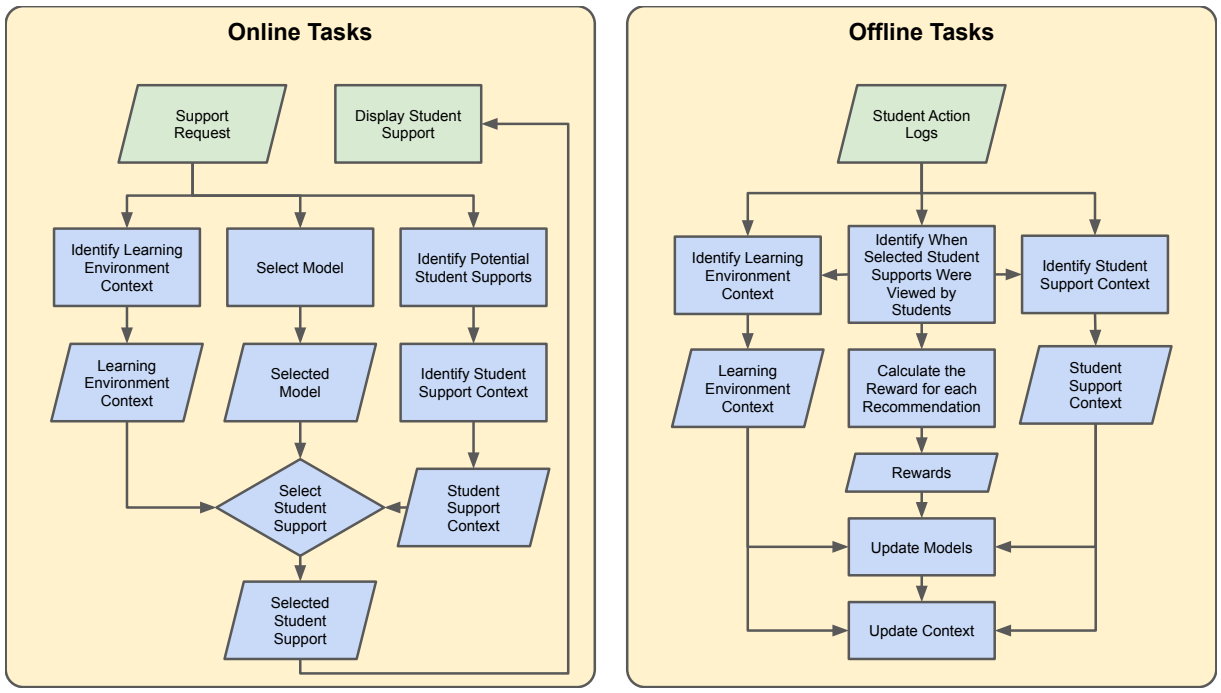

Figure 2: The online (real-time) and offline (nightly) tasks performed by the APLS. Tasks and data from the ASSISTments Tutor are in green, tasks and data from the APLS are in blue.

Algorithm 1 APLS Reward Calculation

```
a is an assignment with n problems
p_i is the ith problem in a
if student s requested tutoring for p_i then
  for m = i + 1, ..., n do
    if p_m is a graded problem then
      if s completed p_m correctly then
        return 1
      else
        return 0
      end if
    end if
  end for
else
  return null
end if
```

reward, which in this case indicates that student was able to get the next graded problem correct, and therefore likely learned from the support. If the student requests support while completing the problem, they are shown the support recommended by the APLS. The first step that the APLS performs when recommending content is to identify all of the supports that are relevant to the request. Each support request contains different IDs, e.g., a user ID, problem ID, assignment ID, and skill ID. These IDs are linked to student supports and determine which student supports are available for recommendation. For example, the most common link is between a student support and a problem ID. These links imply that the student support was written specifically for the problem. A less common link is between a student support and a skill ID, which implies that the student support was written for any problem relevant to a particular skill. This modular linking system allows for educators and researchers to create content related to any aspect of students' learning environment by simply linking the student support to the relevant ID, enabling these student supports to be assessed for their effectiveness and used to further personalize students' learning.

After the potential student supports have been identified, context for the multi-armed bandit algorithm is gathered from the IDs of the request and the potential student supports. Just as each ID in the request can link to a student support, the IDs of the request and the potential student supports can link to an array of features. For example, each problem ID links to an array of performance statistics and HTML-based attributes of that particular problem, and each student support ID links to an array of features on the structure and content of the support. All of the relevant feature arrays are concatenated and become the context used by the multi-armed bandit algorithm. It is useful to note that the context of each potential student support will only differ in the values related to the specific student support, the context related to any ID of the support request will be constant across all potential student supports, as the IDs in the request are not changing between the potential student supports of a single support request.

Once the context of each potential student support is gathered, The APLS selects a multi-armed bandit model using a similar process to the one used to identify potential student supports. First, potential models are selected by identifying models linked to IDs

in the support request, then, one of the potential models is selected at random. Randomly selecting a model facilitates randomized controlled experiments that compare the effectiveness of different models. The first of these experiments is reported on in section 4. Once a model is selected, the context derived from the support request and potential student supports is provided to the model and the model selects the student support it predicts will lead to the highest reward and therefore be most likely to help the student learn. The selected student report is returned to the ASSISTments Tutor and displayed if requested by the student.

3.3 Model Updates

The traditional way to update a multi-armed bandit model is to provide the model with a reward after each recommendation it makes. The model then uses this reward to adjust its internal logic such that it will be more likely to receive higher rewards after future recommendations. However, the models in the APLS can only observe a reward once a student has finished the problem they requested support on, and the order that students complete problems is not necessarily the order that recommendations were made in. Therefore, instead of attempting to update the models in real-time, the models are updated in batches during lulls in user activity.

The APLS tracks which model made each recommendation and the context used to make the recommendation. Each night, after identifying the reward for each recommendation using Algorithm 1, the APLS provides each context-reward pair to the model used to make the recommendation in the order that the recommendations were made. During this process any student support requests sent to the APLS are processed using the models in their states prior to the current update to ensure that there is no downtime or ambiguity in the APLS recommendations. After every model has been updated, the APLS begins to use the updated models to make recommendations.

4 APLS EXPERIMENT

4.1 Experiment Design

To determine the effectiveness of using multi-armed bandit algorithms at scale, a randomized controlled experiment was performed within ASSISTments between November 4th, 2021 and January 2nd, 2022. In this experiment, the APLS randomly selected which of two models it would use to recommend content to students. Students were randomized between the two models each time they started a problem. Therefore, the same student could have received content from both models, but would be unable to determine which model was recommending them content because both models were selecting from the same potential student supports.

The first model used random selection (RS) to select which student support to recommend, and the second model used Beta-Bernoulli Thompson Sampling (BBTS) to select which student support to recommend. BBTS is a simple contextual bandit algorithm for environments with binary rewards. It models the potential reward of each student support as a beta distribution $Beta(\alpha, \beta)$ where α is the number of times the student support was recommended and the model received a reward of 1, and β is the number of times the student support was recommended and the model received a reward of 0. When determining which student support to recommend, a random value is drawn from the beta distribution of each possible student support. The student support corresponding to the highest random value is recommended to the student. BBTS is not aware of the context in which it made recommendations, and learns simply from the rewards that it received in the past [21]. Although BBTS is simple compared to a contextual bandit algorithm, it is a strong baseline from which insight can be gained and advances can be made [6].

Table 1: Results of Thompson Sampling vs. Random Selection

	RS	BBTS
Total Requests	49,740	50,379
Requests with Rewards	40,878 (82.2%)	41,306 (82.0%)
Total Reward	13,529 (33.1%)	13,805 (33.4%)

4.2 Experiment Results

Over the two months that the experiment ran, support was requested by students on about 16.5% of problems. 49,740 support requests were responded to by the APLS using RS, and 50,379 requests were responded to using BBTS. Of all these requests, 82.2% of the RS requests and 82.0% of the BBTS requests did not have a reward due to students receiving support prior to ungraded open ended response questions or on the last problem in their assignment. The average reward received by each model was 0.331 and 0.334 respectively. Table 1 summarizes these findings.

Although BBTS out performed RS, its impact was less than expected based on previous simulations [16]. To understand why BBTS was less effective than expected, it helps to examine the challenges associated with making recommendations within an online learning platform. Firstly, consider the breadth of questions for which supports are being recommended. The BBTS model recommended tutoring for 2,923 different problems, and only recommended each student support an average of 6.5 times. This is very little information to learn from, and thus limited the model's ability to significantly out perform RS. Secondly, consider the way in which students interact with online learning platforms. It is common for a teacher to assign a particular problem set to be completed within the day or week. If an entire class attempts this problem set, and only 16.5% of students request support, half of those supports are going to the RS model, and only four out of five of the support requests can be learned from, than only around 6.5% of students will affect the BBTS model. Furthermore, these students are not coming in a steady stream, but rather in batches of classes. The student supports that work best for one class are not necessary going to be the best for every class. Teachers use different methods of explaining content and the supports that align with the teachers' instruction are more likely to be effective. Therefore, without context of the students' learning environment, the BBTS model could learn relationships from one class that are detrimental to the following class. While the online learning environment made it difficult for BBTS to perform well, a contextual bandit algorithm that can utilize context of the learning environment and student supports to

share insight between student supports can theoretically overcome the shortcomings of BBTS.

5 MOVING FORWARD: DTTS

To overcome the challenges of making recommendations within an online learning platform and to be able to identify opportunities for personalization that can be applied outside of the contextual bandit framework, a recommendation algorithm must be able to accomplish the following:

(1) Apply insight gained from recommending a student support to recommendations of other supports.
(2) Identify generally applicable qualitative interactions between students' learning environment and the effectiveness of different supports.
(3) Learn interpretable insights that can be easily extracted from the model and understood by educators and researchers.

To be able to apply insight gained from recommending a student support to recommendations of other supports, the model must take into account the context of each potential student support when selecting which support to recommend. Models like LinUCB, which use a separate regression for each potential student support would not be able to inform educators of the common trends between student supports without doing additional analysis to combine each regressions' findings [11].

To be able to identify generally applicable qualitative interactions between students' learning environment and the effectiveness of different supports, not only must the model combine context of the learning environment and supports, but it must do so in a non-linear way. Models like Hybrid-LinUCB [11] and Linear Thompson Sampling [1], both popular models that combine context of the learning environment and supports, only form linear relationships between these parameters. A qualitative interaction is a non-linear relationship. To make these models capable of identifying qualitative interactions, one must manually add all the potential interaction terms, and doing so would create thousands of additional variables, making these models slow and prone to over-fitting.

To be able to learn interpretable insights that can be easily extracted from the model and understood by educators and researchers, the model must not abstract the context it is given too severely. There are many neural network based contextual bandit algorithms [19], but the insight gained by these algorithms is represented by weights in neural networks, which are notoriously difficult to interpret as the size of the network increases. Furthermore, difficult to interpret models can lead to skepticism and lack of adoption due to the rising increase in demand for interpretable AI, therefore educators are more likely to accept recommendations from a model that can explain its reasoning [12].

These factors point towards using decision trees to find opportunities for personalization because decision trees are capable of combining the context of students' learning environments and the potential student supports in a non-linear and easily interpretable way. While decision trees are a strong contender for delivering interpretable personalization to students, adapting them to the contextual bandit framework, where they must balance exploration and exploitation to learn how to determine the optimal support for each student, is non-trivial. In the past, researchers attempted to combine decision trees and reinforcement learning by fitting a decision tree each time a recommendation was made on a random subset of the data, which simulated the effect of sampling from a prior distribution [8]. This method has the limitation of needing to retrain the tree to ensure exploration, which is very time-consuming. Another method that attempted to integrate decision trees into the multi-armed bandit framework first explored all possible branches the tree could form, and then selected the branch that led to the highest reward [9]. This method has the limitation of being unable to adapt over time to changing interactions within the context, and requires the exploration of all possible branches, which becomes overwhelmingly time-consuming when there are many possible branches.

Decision Tree Thompson Sampling (DTTS) is a novel decision tree based multi-armed bandit algorithm designed to interpretably recommend support to students within online learning platforms. DTTS integrates decision trees into the multi-armed bandit framework by using the decision tree not as as explicit predictor of the expected reward, but as a model of the prior reward distribution, which can be sampled from to make recommendations via Thompson sampling. As shown in Algorithm 2, given an uninitialized decision tree DT, an empty set of all context observations X, and an empty set of all reward observations R, every n observations, DT is trained on up to m past observations to predict the expected reward given the context. At each time-step t, for all time-steps in T, the leaf node of DT, $l_{x_{t,a}}$, that corresponds to the observed context $x_{t,a}$ is identified for each available action a. In this case, each action is a potential student support. Then, prior distributions are calculated from $R_{l_{x_{t,a}}}$, the subset of R used to create the tree that reached the same leaf node l for each action a out of all K available actions. The prior distribution calculated from $R_{l_{x_{t,a}}}$ is a beta distribution $Beta(\alpha, \beta)$ where $alpha$ is the total number of times the reward was 1 in $R_{l_{x_{t,a}}}$, and β is the number of times the reward was 0 in $R_{l_{x_{t,a}}}$. If the decision tree has not been created yet, then the prior distribution is assumed to be a uniform distribution in the range $[0, 1)$. After determining the prior distributions for each potential action, like in Thompson sampling, each prior distribution is randomly sampled from, and the action corresponding to the random sample $\tilde{\theta}_{t,a}$ with the highest value is chosen. After the action is taken and a reward is observed, the context $x_{t,a}$ and reward r_t are added to X and R respectively. If the length of X and R is greater than m, the oldest observation is removed from X and R. Only storing the most recent m observations helps to keep DT up to date with the current trends in the observations.

6 DTTS SIMULATION

6.1 Simulation Data

The data used to evaluate DTTS came from the ASSISTments TeacherASSIST program. Within ASSISTments, a crowdsourcing effort called TeacherASSIST allowed teachers to create their own student supports for mathematics problems, which were then distributed to all students using ASSISTments. In 2019, a randomized controlled experiment was carried out within ASSISTments in which students were randomized between receiving different

Algorithm 2 Decision Tree Thompson Sampling

DT is an uninitialized CART decision tree
n is an integer. *DT* is trained every *n* observations
X is a set of length *m* that will hold observed contexts
R is a set of length *m* that will hold observed rewards
for t = 1, ..., T **do**
  for a = 1, ..., K **do**
    observe context $x_{t,a}$
    if *DT* is uninitialized **then**
      sample $\tilde{\theta}_{t,a}$ from $[0, 1)$
    else
      sample $\tilde{\theta}_{t,a}$ from $P(R_{l_{x_{t,a}}})$
    end if
  end for
  choose action $a_t = \underset{a}{argmax}\ \tilde{\theta}_{t,a}$
  observe reward r_t
  add $x_{t,a}$ and r_t to *X* and *R* respectively
  if Length of $X > m$ **then**
    remove the oldest observation from both *X* and *R*
  end if
  if $t \geq n$ and $t \pmod n = 0$ **then**
    train *DT* to predict *R* using *X*
  end if
end for

crowdsourced supports. This study found that providing students with crowdsourced supports led to greater learning gains than when students were only given the answer after requesting support [15]. In 2020, the 2019 study was repeated and the same results were found. The data collected from the TeacherASSIST program contains 399,869 instances of a struggling student requesting support, being given one of multiple possible crowdsourced student supports, and then having the opportunity to answer a graded problem within the same assignment. 1,946 teachers, 5,635 classes, 27,712 assignments, 62,056 students, 5,470 mathematics problems, and 13,394 different student supports are contained within this data. For each teacher, class, assignment, user, problem, and next problem, prior performance statistics are available for every support request. Various features of the problem sets, problems, and student supports are also available. Additionally, in order to make the insight gained from these features more interpretable, for this simulation, each continuous feature was converted into a binary indicator of whether the value was above average or not. The full dataset and a description of all of its contents is hosted by the Open Science Foundation at https://osf.io/9pgv5/. In total, 98 of the available features were used in this analysis, specifically the features focused on prior statistics and aspects of the students, problems, and student supports. This data is ideal for determining the effectiveness of different recommendation algorithms within online learning platforms because it contains real information on the learning gains of thousands of students after receiving one of thousands of supports, and the features are nearly identical to the context used by the APLS described in Appendix A.

6.2 Simulation Design

The data from TeacherASSIST can be sampled from to simulate a mutli-armed bandit algorithm operating within an online learning platform. This can be achieved using the following simulation strategy:

(1) Initialize a multi-armed bandit algorithm that may take features of the student, problem, and potential student supports as context and uses the reward metric described by Algorithm 1.
(2) Randomly sample with replacement a single instance of a student receiving support from all the TeacherASSIST data.
(3) Use the bandit algorithm to recommend which tutoring, from all the possible student supports, the student in the random sample should receive.
(4) If the recommended support was the support that the student actually received, then update the bandit algorithm using the calculated reward, otherwise ignore this sample and go back to step 2.
(5) Repeat steps 2-4 as many times as desired to simulate the contextual bandit algorithm making multiple sequential recommendations.

6.3 Learning Impact

The process described in Section 6.2 can be used to answer the question "How would students' learning gains have been different if a multi-armed bandit algorithm besides BBTS was used to recommend student supports?". To evaluate the effectiveness of DTTS, three different simulation studies were run in which DTTS was compared to RS, BBTS [21], and Linear Thompson Sampling (LTS) [1]. In all three simulations, DTTS used a CART decision tree [4] that used Gini Impurity as its split criteria, and the ϵ and δ parameters of LTS were set using Theorem 1 and Remark 2 of [1] to obtain a 95% chance of having an $\tilde{O}(d^2\sqrt{T})$ regret bound. The first simulation used the process described in Section 6.2 to sample from all the TeacherASSIST data 376,674 times, which is how many recommendations with reward were collected by TeacherASSIST in the year 2020. This gives insight into how DTTS would have performed compared to random selection and popular multi-armed bandit algorithms over the course of a full year.

The second and third simulations investigated how capable DTTS is of generalizing its insight to new content. The second simulation is similar to the first, but it divided the data into groups, where each group contained unique students. Each group was sampled from approximately 1,721 times, which is the average number of recommendations with reward made in a single day, before moving on to the next group. This simulation helped evaluate DTTS's ability to learn contextual insight beneficial to new students because each simulated day, the students were unique. Therefore DTTS would have to gain generalizable insight from students' context to be able to make helpful recommendations. The third simulation was similar to the second except it divided the data into groups of unique problems and student supports. This simulation helped evaluate DTTS's ability to learn generalizable insight applicable to new problems and supports. The cumulative reward over 376,674 observations and the percent of recommendations with a reward of 1 for every model in each simulation is shown in Table 2.

Table 2: Cumulative Rewards and % Maximum Reward for all Simulations

Simulation Description	RS	BBTS	LTS	DTTS
All Data	127,302 (33.8%)	133,601 (35.5%)	126,449 (33.6%)	**146,983 (39.0%)**
Student Groups	127,024 (33.7%)	**134,615 (35.7%)**	127,080 (33.7%)	134,073 (35.6%)
Problem Groups	127,000 (33.7%)	134,028 (35.6%)	127,274 (33.8%)	**134,714 (35.8%)**

Table 2 shows that, overall, DTTS not only outperformed BBTS, which is currently used in the APLS, but also outperformed LTS, a well established contextual bandit model. In the student groups simulation, BBTS sightly out performed DTTS. This implies that while DTTS is capable of generalizing findings between students, these findings do not provide more insight than modeling the reward of each support independently. In the problem groups simulation, DTTS was again able to out-perform all other models, which implies that DTTS can learn relationships that generalize across problems and tutoring. Across all three simulations, DTTS demonstrated that the contextual insight it learned was applicable both to new students, and new problems and student supports. Overall, it seems that interactions between features of the problems and student supports are more generalizable than interactions with features of the students.

The difficulty of using multi-armed bandit algorithms in an online learning environment is made clear by LTS's inability to outperform BBTS. Although this simulation was focused on evaluating the effectiveness of DTTS, this is a valuable finding because it implies that the quality of the student support is dependent on the learning environment. LTS can only model how much each feature of student supports influences the likelihood of the student getting the next graded problem correct. Without interactions between context of the student supports and context of the learning environment, it struggles to gain insight. For example, LTS can only learn that longer student supports have an overall positive or overall negative effect on students' learning, but BBTS can learn that for one particular problem, a longer support is better, and for a different problem, a shorter support is better, even though it does not know the length of the student supports. Without being able to generalize this finding across multiple supports, BBTS still out-performs LTS. This finding implies that DTTS outperformed BBTS and LTS because it could both utilize non-linear relationships, and generalize these relationships to previously unseen students, problems, and student supports.

6.4 Personalization Insight

Although we have shown that DTTS is more capable of increasing students' learning gains than other multi-armed bandit algorithms, there is no guarantee that it does so by identifying and taking advantage of qualitative interactions between students' learning environment and the potential student supports. To investigate if this is the case, the simulation strategy described in Section 6.2 was again used to simulate a year of DTTS based recommendations, but the simulation only used a random half of the TeacherASSIST data. After the simulation finished, the total amount that each feature of the context contributed to the reduction of Gini Impurity in the final decision tree formed by DTTS was compared to the strength of the qualitative interactions that existed in the other half of the TeacherASSIST data not used in the simulation. To make this comparison, the first step was to use Equation 1 to fit a model for each of the 2,001 possible qualitative interactions between the student supports and the learning environment, where x_1 is a feature of the student supports, x_2 is feature of the learning environment, and y is the reward calculated using Algorithm 1.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 \oplus x_2) \tag{1}$$

For there to be a qualitative interaction, the effect of x_1 when $x_2 = 0$, $\text{eff}_{x_1|x_2=0}$, must have the opposite sign as the effect of x_1 when $x_2 = 1$, $\text{eff}_{x_1|x_2=1}$, or in other words, the product of the two effects must be less than zero. This can be determined by whether or not β_3^2 is greater than β_1^2, using the following logic.

$$E[y|x_1 = 0, x_2 = 0] = \beta_0$$
$$E[y|x_1 = 1, x_2 = 0] = \beta_0 + \beta_1 + \beta_3$$
$$E[y|x_1 = 0, x_2 = 1] = \beta_0 + \beta_2 + \beta_3$$
$$E[y|x_1 = 1, x_2 = 1] = \beta_0 + \beta_1 + \beta_2$$

$$\text{eff}_{x_1|x_2=0} = E[y|x_1 = 1, x_2 = 0] - E[y|x_1 = 0, x_2 = 0]$$
$$\text{eff}_{x_1|x_2=0} = (\beta_0 + \beta_1 + \beta_3) - (\beta_0) = \beta_1 + \beta_3$$
$$\text{eff}_{x_1|x_2=1} = E[y|x_1 = 1, x_2 = 1] - E[y|x_1 = 0, x_2 = 1]$$
$$\text{eff}_{x_1|x_2=1} = (\beta_0 + \beta_1 + \beta_2) - (\beta_0 + \beta_2 + \beta_3) = \beta_1 - \beta_3$$

$$(\beta_1 + \beta_3)(\beta_1 - \beta_3) < 0$$
$$\beta_1^2 - \beta_3^2 < 0$$
$$\beta_1^2 < \beta_3^2$$

Once it has been determined whether or not there is a qualitative interaction between two features, the absolute value of the t-value of β_3 can be used to measure the strength of the qualitative interaction in a way that takes into account the magnitude and the standard error of the β_3 coefficient.

After the presence and strength of each potential qualitative interaction was determined, the strength was compared to the product of the total reduction of Gini Impurity that each of the features in the interaction is responsible for in the decision tree formed by DTTS. If the magnitude of the t-value is correlated with the product of the total reductions in Gini Impurity, this implies that stronger qualitative interactions are used more by DTTS. This helped determine whether the insight learned by the decision tree involved personalization, and if that insight was applicable to new

Table 3: Correlation Between Qualitative and Non-Qualitative Interactions and the Product of the Total Reduction in Gini Impurity of Each Feature in the Interaction Determined Using the Decision Tree Formed by DTTS

	Total #	*rho*	*p*-value
Qualitative Interactions	823	0.11	0.002
Non-Qualitative Interactions	1178	0.06	0.052

data or just over-fit assumptions based on the data used to create the decision tree.

Using the above methodology, the correlation between the magnitude of each potential qualitative interaction's t-value and the product of the total reductions in Gini Impurity of each feature of the interaction in the DTTS decision tree was determined using Spearman's rank correlation coefficient [22]. The results of this, shown in Table 3 demonstrate that when there is a qualitative interaction between two features, the decision tree utilizes the features more as the strength of the qualitative interaction increases, and when there is no qualitative interaction, there is no correlation with how valuable the features are to the decision tree. This implies that the decision tree formed by DTTS is taking advantage of qualitative interactions to personalize students' learning. Due to the interpretability of decision trees, it is possible to search the tree, identify these interactions, and use them to inform the design of curricula with the intent of personalizing students' education.

7 LIMITATIONS AND FUTURE WORK

The APLS, while a promising tool for personalizing students' online education, has yet to reach its full potential. While the process of recommending content is modular, it relies on having many optional supports available. Personalized learning relies on having enough content such that each student can be delivered support suitable for their needs. In the future, collecting more student support messages, either through crowdsourcing or algorithmic generation, will be key in ensuring that any online learning platform can provide quality personalization. Additionally, the ability to collect relevant context on the student supports and learning environment is imperative to the success of personalized learning at scale. The APLS currently collects context on the prior statistics, format, and HTML-based attributes of the students, problems, and student supports. This can be expanded, not only to include more features from within the online learning platform, such as teachers' behavioral patterns, but also to include features such as the sentiment or tone of the student supports or the emotional state of the students. These features, which can be collected with a variety of different algorithms, e.g. [2, 3], would help the APLS make more informed and insightful recommendations.

The APLS has only been tested using BBTS, a simple multi-armed bandit algorithm to recommend support to students. However, DTTS was able to overcome the difficulties of recommending content within an online learning platform and outperformed both BBTS and LTS in simulation. While the future of DTTS is bright, more work needs to be done to confirm the findings of this simulation. The next step is to integrate DTTS into the APLS and empirically measure its impact on learning at scale. Additionally, while DTTS was the most effective algorithm, it was also the only algorithm that needed to record previous observations. Both BBTS and LTS need to only store statistics on the previous observations, making them much more memory efficient. Moving forward, experimenting with using Hoeffding Trees instead of CART Decision Trees could reduce the memory cost. Hoeffding Trees are designed to use each observation as input only once and store statistics on each observation instead of a history of observations [7]. Beyond exploring improvements to DTTS, the purpose of using decision trees was to ensure interpretability. Therefore, when DTTS is integrated into online learning platforms, work should be done to make the recommendations made by DTTS as transparent as possible. Creating a user interface to help educators and researchers understand what qualitative interactions DTTS is taking advantage of when recommending content would both facilitate adoption of DTTS and inform personalized learning pedagogy.

8 CONCLUSION

This work presented the Automatic Personalized Learning Service (APLS), a novel infrastructure for personalized learning within ASSISTments, an online learning platform with around 300,000 active users. An empirical study was performed and it was demonstrated that using the APLS to recommend support for struggling students with Beta-Bernoulli Thompson Sampling (BBTS), a common and effective multi-armed bandit algorithm, was only slightly better that selecting from the optional relevant student supports at random. Further investigation revealed that the lack of significant improvement was due to the breadth of problems for which support needed to be recommended and the sparsity in opportunity to recommend the same content multiple times. These shortcomings prompted the creation of Decision Tree Thompson Sampling (DTTS), a novel multi-armed bandit algorithm for recommending content that combines the interpretability and non-linearity of decision trees with Thompson sampling's proven approach to the exploitation-exploitation trade-off. DTTS was shown in simulation to outperform both BBTS and Linear Thompson Sampling (LTS), demonstrating its ability to learn generalizable insights into how to effectively personalize learning for students, problems, and student supports that it had no prior exposure to. Additionally, using DTTS to simulate recommending student supports, then correlating the importance of each feature, as determined by the decision tree made by DTTS, to the magnitude of every potential qualitative interaction in a separate dataset found that the importance of the features correlated with the strength of their qualitative interaction. Implying that the insight gained by DTTS is applicable to new content and relies on the identification of qualitative interactions, which are essential for personalization. Moving forward, DTTS will be integrated into the APLS in ASSISTments where it can begin to personalize learning for thousands of students across the world and other platforms can begin to integrate the APLS framework using DTTS.

ACKNOWLEDGMENTS

We would like to thank NSF (e.g., 2118725, 2118904, 1950683, 1917808, 1931523, 1940236, 1917713, 1903304, 1822830, 1759229, 1724889,

1636782, 1535428), IES (e.g., R305N210049, R305D210031, R305A170-137, R305A170243, R305A180401, R305A120125), GAANN (e.g., P200A180088 P200A150306), EIR (U411B190024), ONR (N00014-18-1-2768) and Schmidt Futures.

REFERENCES

[1] Shipra Agrawal and Navin Goyal. 2013. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*. PMLR, 127–135.

[2] María Lucía Barrón-Estrada, Ramon Zatarain-Cabada, Raúl Oramas-Bustillos, and Francisco González-Hernández. 2017. Sentiment analysis in an affective intelligent tutoring system. In *2017 IEEE 17th international conference on advanced learning technologies (ICALT)*. IEEE, 394–397.

[3] Anthony F Botelho, Ryan S Baker, and Neil T Heffernan. 2017. Improving sensor-free affect detection using deep learning. In *International conference on artificial intelligence in education*. Springer, 40–51.

[4] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 2017. *Classification and regression trees*. Routledge.

[5] Monica Bulger. 2016. Personalized learning: The conversations we're not having. *Data and Society* 22, 1 (2016), 1–29.

[6] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems* 24 (2011), 2249–2257.

[7] Pedro Domingos and Geoff Hulten. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 71–80.

[8] Adam N Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. 2017. A practical method for solving contextual bandit problems using decision trees. *arXiv preprint arXiv:1706.04687* (2017).

[9] Raphaël Féraud, Robin Allesiardo, Tanguy Urvoy, and Fabrice Clérot. 2016. Random forest for the contextual bandit problem. In *Artificial intelligence and statistics*. PMLR, 93–101.

[10] Chen-Lin C Kulik and James A Kulik. 1982. Effects of ability grouping on secondary school students: A meta-analysis of evaluation findings. *American educational research journal* 19, 3 (1982), 415–428.

[11] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

[12] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2021. Explainable ai: A review of machine learning interpretability methods. *Entropy* 23, 1 (2021), 18.

[13] Korinn Ostrow, Neil Heffernan, and Joseph Williams. 2017. Tomorrow's EdTech today: Establishing a learning platform as a collaborative research tool for sound science. *Teachers College Record* 119, 3 (2017), 1–36.

[14] John F Pane, Beth Ann Griffin, Daniel F McCaffrey, and Rita Karam. 2014. Effectiveness of cognitive tutor algebra I at scale. *Educational Evaluation and Policy Analysis* 36, 2 (2014), 127–144.

[15] Thanaporn Patikorn and Neil T Heffernan. 2020. Effectiveness of crowd-sourcing on-demand assistance from teachers in online learning platforms. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 115–124.

[16] Anna Rafferty, Huiji Ying, Joseph Williams, et al. 2019. Statistical consequences of using multi-armed bandits to conduct adaptive educational experiments. *Journal of Educational Data Mining* 11, 1 (2019), 47–79.

[17] Leena M Razzaq and Neil T Heffernan. 2009. To Tutor or Not to Tutor: That is the Question.. In *AIED*. 457–464.

[18] Mohi Reza, Juho Kim, Ananya Bhattacharjee, Anna N Rafferty, and Joseph Jay Williams. 2021. The MOOClet Framework: Unifying Experimentation, Dynamic Improvement, and Personalization in Online Courses. In *Proceedings of the Eighth ACM Conference on Learning@ Scale*. 15–26.

[19] Carlos Riquelme, George Tucker, and Jasper Snoek. 2018. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127* (2018).

[20] Jeremy Roschelle, Mingyu Feng, Robert F Murphy, and Craig A Mason. 2016. Online mathematics homework increases student achievement. *AERA open* 2, 4 (2016), 2332858416673968.

[21] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. 2017. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038* (2017).

[22] Charles Spearman. 1961. The proof and measurement of association between two things. (1961).

A APLS FEATURE DESCRIPTIONS

For users, summary statistics based on all prior problems completed by the user are calculated. These same summary statistics are calculated for problems based on all prior instances of a student attempting the problem. These summary statistics, listed below, make up 20 of the context features.

- **total_assignments_completed**: For users, the total number of assignments completed previously. For problems, The total number of assignments with the problem in it completed previously.
- **total_problems_completed**: For users, the total number of problems completed previously. For problems, The total number of times the problem was completed previously.
- **assignment_completion_percentage**: For users, the percent of previously started assignments that were completed. For problems, the percent of previously started assignments with the problem in it that were completed.
- **problem_completion_percentage**: For users, the percent of previously started problems that were completed. For problems, the percent of previous times the problem was started that is was also completed.
- **median_time_on_task**: For users, the median time on task spent on problems. For problems, the median time on task spent by students on the problem.
- **median_first_response_time**: For users, the median time spent before submitting an answer or requesting tutoring when completing a problem. For problems, the median time spent before submitting an answer or requesting tutoring for students completing the problem.
- **average_correctness**: For users, the fraction of times they got previously completed problems correct on their first attempt. For problems, the fraction of times that students attempting the problem got it correct on their first attempt.
- **average_attempt_count**: For users, the average number of attempts per problem on previously completed problems. For problems, the average number of attempts for students that completed the problem.
- **average_hint_count**: For users, the average number of hints used per previously attempted problem. For problems, the average number of hints used by students that previously completed the problem.
- **average_first_action_answer**: For users, the fraction of times that the student attempted to answer the problem before requesting tutoring. For problems, the fraction of times that students completing the problem attempted to answer the problem before requesting tutoring.

One-hot encoded categorical variables for problem answer type, grade level, and subject are also included in the context of the APLS. Descriptions of the 10 answer types are listed below.

- **problem_type_1**: Multiple Choice
- **problem_type_2**: Check All That Apply
- **problem_type_3**: Place Items In Order
- **problem_type_4**: Exact Match (case sensitive)
- **problem_type_5**: Legacy Algebraic Expression, e.g., z = 2y
- **problem_type_11**: Exact Match (ignore case)
- **problem_type_13**: Number, e.g., 93
- **problem_type_14**: Numeric Expression, e.g., 3 + 2 * 4
- **problem_type_15**: Exact Fraction, e.g., 3/2
- **problem_type_17**: Algebraic Expression, e.g., z = 3x + 2y

The 15 grade level features and 32 subject features are described by the Common Core State Standards for Mathematics, which can be found at http://www.corestandards.org/Math/. The first and second section of the Common Core Skill Code correspond to the grade level and subject respectively. For example, for the skill code **7.RP.A.2.d**, the grade level is **7** and the subject is **RP**.

Lastly, problems and student supports have context used by the APLS corresponding to their HTML structure. Student supports have more structural features than problems. Therefore, in the following list, every feature exists for student supports and starred features also exist for problems.

- **student_support_content_creator_id**: The ID of the creator of the student support.
- **student_support_hint**: A binary indicator of whether or not the student support is a hint.
- **student_support_explanation**: A binary indicator of whether or not the student support is an explanation.
- **student_support_message_count**: The number of messages contained within the student support.
- ***_text_length**: The character count of all the text in the problem or student support.
- ***_contains_video**: A binary indicator of whether or not the problem or student support contains a video.
- ***_contains_image**: A binary indicator of whether or not the problem or student support contains an image.
- ***_contains_link**: A binary indicator of whether or not the problem or student support contains a link.
- ***_color_use**: A binary indicator of whether or not the problem or student support uses different text colors.
- ***_font_use**: A binary indicator of whether or not the problem or student support uses different text fonts.
- ***_text_size_use**: A binary indicator of whether or not the problem or student support uses different text sizes.