Predicting Challenge Outcomes for Students in a Digital Game for Learning Genetics

Ziwei Wu[1], Bradford Mott[1], Wookhee Min[1], Robert Taylor[1], Danielle Boulden[2], Trudi Lord[3], Frieda Reichsman[3], Chad Dorsey[3], Eric Wiebe[2], and James Lester[1]

[1]Department of Computer Science, North Carolina State University, Raleigh, NC, 27695, USA
[2]Department of STEM Education, North Carolina State University, Raleigh, NC, 27695, USA
[3]Concord Consortium, Concord, MA, 01742, USA
[1]{zwu17, bwmott, wmin, rgtaylor, lester}@ncsu.edu
[2]{dmboulde, wiebe}@ncsu.edu
[3]{tlord, freichsman, cdorsey}@concord.org

ABSTRACT

In recent years, digital games for learning have shown significant potential for creating engaging and effective student learning experiences. A common gameplay design used by many digital games for learning is providing students with a series of challenges featuring varying levels of difficulty. Identifying whether students will struggle on certain challenges is a key task in these environments because it could support adaptively adjusting difficulty levels and providing immediate assistance to students. In this paper, we present a data-driven approach to modeling students' gameplay behaviors with challenges in an open-ended learning environment for introductory genetics. Challenge outcome prediction models utilize students' observed gameplay behaviors with previous challenges to classify students' performance on the next challenge into two categories: quit or complete. We build machine learning models for predicting students' gameplay performance by taking advantage of a corpus of 633 students' in-game behaviors from Geniventure, a digital game for learning genetics. We compare the accuracy of the models to gain insights into which models perform best for this prediction task. Results show that support vector machine (SVM) models produce the overall best performance in predicting gameplay outcomes for challenges.

Keywords

Game-based Learning, Digital Games, Quit Prediction, Adaptive Instruction

1. INTRODUCTION

Recent years have seen growing interest in digital games for learning because of their potential for creating engaging and effective student learning experiences [7][27]. Researchers have investigated digital games for learning in a wide array of educational domains, including mathematics [28][18], computational thinking [4][14], and science [30][1][12]. Well-designed digital games for learning must carefully balance students' engagement in the gameplay experience with a focus on the overall learning objective [12]. A common gameplay design used by many digital games for learning is providing students with a series of challenges featuring varying levels of difficulty.

Successful game-based learning experiences that simultaneously promote engagement while improving learning outcomes are possible, but they must be carefully designed [29]. Researchers suggest that well-designed educational games should provide students' with just-in-time support while focusing their gameplay at the edge of their abilities, ensuring students remain challenged throughout the learning experience [9]. To achieve this goal, digital games for learning should have the ability to detect when students are struggling and take action to tailor their learning experience to provide appropriate levels of challenge [31]. With recent advances in machine learning techniques, data-driven approaches using students' in-game behaviors have enabled the automatic assessment of students' evolving competence [25][35] and the modeling of important learning phenomenon, including mind wandering [6][15] and wheel spinning [3]. One interesting avenue of research in student modeling is to examine students' quitting behaviors associated with negative learning outcomes [24][17]. It is particularly important to design robust predictive models for students' quitting behaviors, since a digital game with this functionality can, in advance, guide students from undertaking a challenge that is beyond the learners' capabilities at that moment.

The goal of this paper is to detect whether a student is likely to quit an upcoming challenge in a digital game for learning. In this work, we present a data-driven approach to modeling students' quitting behaviors in an open-ended digital game for learning genetics, Geniventure [23]. In Geniventure, students learn genetics by engaging in challenges of varying difficulty levels. Within the game, although students are encouraged to solve challenges in a linear manner, they can autonomously choose which challenge they play as well as to leave a challenge prior to completing it. During gameplay, students' gameplay trajectory and their detailed in-game actions were recorded as trace data logs. Fine-grained and descriptive features were engineered to effectively capture salient learning trajectories which are useful to predict quitting behaviors per challenge. To incorporate students' gameplay trajectory information into the model, *n*-gram features, which are a contiguous subsequence of *n* actions from a sequence of actions, were employed. A suite of machine-learned predictive models was trained using the extracted features to better understand which approach offers the best predictive performance. We compare the performance of the different machine learning algorithms under two different feature sets in order to gain insights into which learning algorithms and features perform best for this task.

2. RELATED WORK

Engagement is a key component of successful learning. Detecting disengagement has been of great interest to educational researchers [6][15][24][17]. Engagement is often viewed as encompassing three primary components: emotional, behavioral, and cognitive [8]. Disengagement detectors usually target elements of these three components, by making inferences about students' emotional or cognitive state based on their behaviors. In contrast to models of detecting disengagement whose ground truth labels are collected by humans in various manners such as field observations, self-reports, and retrospective judgement, students' quitting behaviors can be directly identified from learning environment trace data [2].

There is considerable research on predicting students' quitting behaviors in the context of massive open online courses (MOOCs). In MOOCs, students' quitting behavior is usually referred to as dropout, which indicates situations where a student registers for a course and makes initial effort on the course activities but eventually quits before completing the course [21]. Much of the work on dropout prediction in MOOCs focuses on developing features from students' behaviors and engagement patterns to help improve prediction [13]. For instance, Kloft et al. predict dropout from only click-stream data using a support vector machine (SVM) [20]. Halawa et al. study early dropout prediction using student activity features capturing lack of ability or interest [11]. Taylor et al. employed crowd-sourced feature engineering from raw trace data collected from thousands of students to predict dropout using logistic regression [33]. In more recent studies, deep learning techniques have been employed, achieving high performance on predicting dropout by taking advantage of large amounts of student data. Yuntao et al. proposed a composite model to infer students' dropout behaviors based on a historical log of their learning activities, including interaction with video lectures, participation in discussion forums, and performance on assignments. The authors employed a stacked sparse autoencoder model combined with a recurrent neural network model to learn high-level representations of input features and implemented an SVM for final classification of dropout [21].

Beyond dropout prediction in MOOCs, previous work has also explored various data-driven approaches to predicting students' quitting behavior in learning environments. Mills et al. developed detectors to predict students' behavioral disengagement through their quitting behaviors while reading instructional texts. Supervised machine learning algorithms were used to predict whether students would quit reading an upcoming text based on features extracted from reading behaviors on previous texts [24]. Karumbaiah et al. presented a quitting prediction model for students playing an educational game called Physics Playground. Gradient boosting classifiers were trained using a set of engineered features from students' interaction data. The features were of different levels of granularity and used to train both level-specific models and level-agnostic models to predict students' quitting behavior on levels within the game. Level-agnostic models were found to provide better predictive performance [17]. Similar to the digital learning environment in [17], our digital learning environment also contains different challenge levels. In our work, we build a single integrated model to predict students' quitting behaviors for all challenges. Different from the level-agnostic model in [17], our model uses *n*-gram features to incorporate students' historical gameplay information, while they calculated accumulated features to summarize historical data. Our work also investigates the performance of different machine learning algorithms for this task.

3. EXPERIMENTAL SETUP

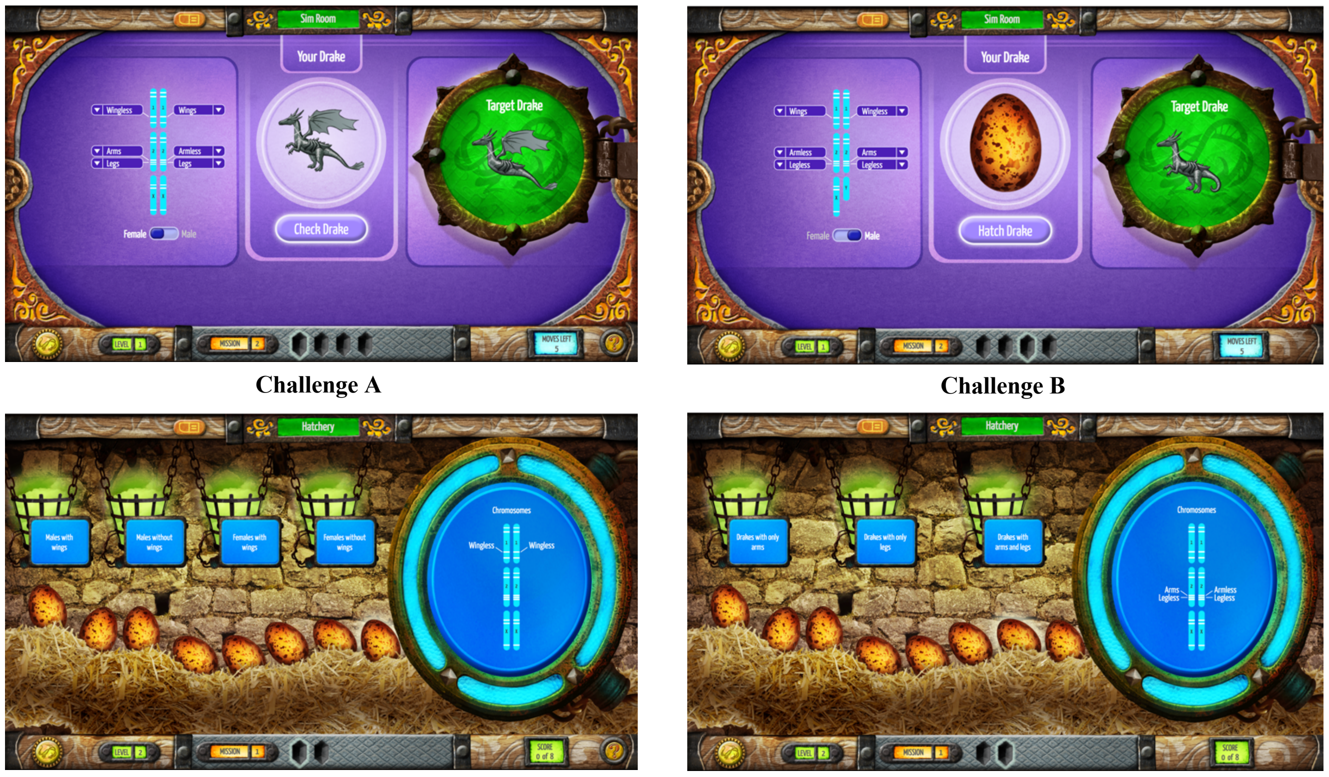
We investigate our approach of modeling students' outcomes on challenges with data collected from high school and middle school students. In this section, we describe the digital game for learning, its problem-solving challenges, and the dataset generated from students' interactions with the learning environment.

3.1 Geniventure

Geniventure is a digital game environment developed for middle school and high school students (11 ~ 18 years old) to learn genetics. The design of the game was guided by core ideas in genetics and science practices aligned with the Next Generation Science Standards [32], a set of science education standards developed in the United States. In the game, students learn concepts in genetics by completing problem-solving challenges centered around breeding dragons [23]. The game consists of 6 levels and over 60 problem-solving challenges of varying levels of difficulty. Each challenge is designed around one or more genetic concepts with the same concept potentially appearing across multiple challenges. Problem-solving challenges within the game appear in a variety of types. When students launch the game, they have the option to decide which challenge to begin with and they are free to quit a challenge at any time during the game. If students finish a challenge, the game rewards them with a colored crystal based on their efficiency in solving the challenge. Students can then decide whether to try the same challenge again or move on to another challenge.

The goal of this work is to build models that can accurately predict students' outcomes on a problem-solving challenge before they begin the challenge. We focus on challenges from the first two levels in Geniventure, which test four fundamental concepts in genetics: *simple dominance*, *recessive traits*, *sex determination*, and *genotype-to-phenotype mapping*. These four concepts are critical for students to understand more complex genetic phenomena covered in later challenges. In this work, four distinct challenges are noted as *Challenge A*, *Challenge B*, *Challenge C*, and *Challenge D*, respectively (Figure 1). Each of these challenges cover all of the four fundamental concepts. Because of different task settings, there are differences in challenges with respect to the difficulty of solving them. We observed that, *Challenge A* and *Challenge B* are relatively easier than *Challenge C* and *Challenge D* based on students' success rate of completing the challenges (Table 1).

In *Challenge A* and *Challenge B* (Figure 1, Top), students are shown a target dragon with certain traits on the right side of the screen. On the left side of the screen, the game provides students with options to manipulate the alleles of the dragon they are creating. Students have options to set alleles to a dominant gene or recessive gene that determine the traits of their dragon. The goal of these two challenges is to create a dragon with the same traits as the target dragon. Both *Challenge A* and *Challenge B* follow this mechanic, but the visibility of the dragon being created varies between the two. In *Challenge A*, students immediately see the changes to the dragon they are creating as the alleles are manipulated. However, in *Challenge B,* the dragon they are creating is hidden until students have selected the alleles and requests the dragon to be hatched. To successfully complete these problem-solving challenges, students must understand several genetic concepts and be able to infer the phenotype of their dragons from its genotype. At the start of the challenge, the game randomly generates an initial set of alleles that require the student to make at least one selection for each allele to achieve the target trait. The "Moves Left" indicator in the lower right corner of the game's display is initialized with the minimum number of allele changes needed to generate the target dragon from the initial configuration of alleles. The indicator will decrement each time a student makes a change to the alleles. Once students feel they have the correct genotype, they click the "Check" or "Hatch" button to submit their answer. If the dragon they create matches the target dragon, the challenge is successfully completed. Otherwise, the game provides the student with feedback and allows them to continue to make further changes to the alleles until they quit or successfully complete the challenge. In *Challenge C* and *Challenge D*, students must sort eggs into the correct basket based on their traits. Students can receive information about the genotype of each egg using the scope on the right side of the screen.



Challenge A **Challenge B**

Challenge C **Challenge D**

Figure 1. Four problem-solving challenges in the Geniventure learning environment

3.2 Dataset

In this work, we analyzed data from 654 students (299 female, 305 male, and 50 unreported) from seven high schools (six public schools, one private school) and one public middle school located in the Middle to Northern Atlantic coast of the United States. Among the students, 100 of them reported being in 6th to 8th grade, 544 reported being in 9th to 12th grade, and 10 students did not report their grade level. This data was conducted during a teacher-led classroom implementation of Geniventure where students played the game during class over the course of several days. Before playing the game, students took a pre-test consisting of 24 questions related to the genetic concepts covered in the game. Five of these questions assessed the genetic concepts in the four types of challenges described earlier. Once gameplay concluded, students took a post-test which was identical to the pre-test. Both the pre-test and post-test were online surveys accessible through the same online portal as the game. We focus on students' performance on the five questions aligned to the four previously identified concepts being examined. Results from a paired t-test on students' knowledge pre-test (*M*=2.971, *SD*=1.52) and post-test (*M*=3.878, *SD*=1.40) revealed a significant improvement from pre-test to post-test ($t(653) = 15.85$, $p < 0.001$, Cohen's $d = 0.621$).

4. METHODS

We first describe students' quitting behaviors that occur in the game and then discuss our feature engineering process.

4.1 Students' Quitting Behaviors

We define "quitting a challenge" as being anytime a student leaves a challenge without successfully completing it. In Geniventure, a challenge is considered as successfully completed only when the crystal awarding screen appears. For *Challenge A* and *Challenge B*, students will be directed to the crystal awarding screen only after they reach a correct answer. For *Challenge C* and *D*, it will be after they sort each of the 8 eggs into a basket no matter whether the basket-egg match is correct or not. We identified three types of quitting behaviors: (1) A student starts a challenge but leaves the challenge before making any moves; (2) A student starts a challenge, makes a few moves, but leaves the challenge prior to submitting an answer; (3) A student starts a challenge, makes some moves, and submits at least one wrong answer before leaving the challenge.

Out of 654 students, we removed 21 students who played less than two challenges for our analysis, since they would not provide sufficient details to infer students' quitting behaviors. In the dataset, the challenges were played 6,568 times by 633 students in total (*M* = 10.38). Among all these attempts, *Challenge A* was played 1,983 times, *Challenge B* was played 2,795 times, *Challenge C* was played 1,116 times, and *Challenge D* was played 674 times. The overall class label distribution in the dataset is highly imbalanced, having 17.1% quit and 82.9% completed. Table 1 shows the summary statistics for challenges.

We examined students' trace data logs, and observed that it is common for a student to play the same challenge repeatedly or revisit an easier challenge after a few unsuccessful attempts on a more difficult challenge. To represent a student's trajectory of playing a sequence of challenges, we use $T_i = \{C_1, C_2, \ldots, C_N\}$, where N denotes the number of challenge attempts, and each $C_k (1 \leq k \leq N)$ in the trajectory (T_i) is the *k*-th challenge student

Table 1. Summary statistics for each type of challenge

	Quit	Complete	Total
Challenge A	251 (12.7%)	1,732 (87.3%)	1,983
Challenge B	330 (11.8%)	2,465 (88.2%)	2,795
Challenge C	376 (33.7%)	740 (66.3%)	1,116
Challenge D	166 (24.6%)	508 (75.4%)	674
Total	1,123 (17.1%)	5,445 (82.9%)	6,568

i played. Note that $C_k \in \{A,\ B,\ C,\ D\}$ where *A, B, C,* and *D* are the four types of challenges and the subscript k denotes the order the challenges were played. The length N of the trajectory varies between students. Likewise, there is a series of outcome labels $O_i = \{L_1,\ L_2, \dots,\ L_N\}$ that correspond to the trajectory of student *i*, where $L_k \in \{quit,\ completed\}$ denotes whether the student quit or completed challenge C_k. Our goal is to induce predictive models which can dynamically predict students' quitting or completing behavior for the next possible challenges they will play, utilizing their observed previous gameplay trajectory (e.g., predicting L_4 utilizing $C_1, C_2, C_3, L_1, L_2, L_3$). In other words, as soon as the student finishes C_k, our model performs predictions of whether the student will quit or complete challenge C_{k+1}.

4.2 Feature Engineering

Feature engineering is a critical step in building models to predict students' quitting behaviors. Feature engineering converts students' raw, low-level interaction data, and pre-learning measures into a trainable format. In our predictive model, each student's problem-solving trajectory for a *Challenge C* is represented with a set of *M* features $F_c = \{f_c^1,\ f_c^2, \dots,\ f_c^M\}$ that captures (1) the challenge type, (2) the student's pre-test score on the concept knowledge, and (3) a sequence of actions the student took while interacting with the challenge. All of these features are designed to be generalizable to other digital games for learning.

First, one feature is created to represent the challenge, which in this work is one of the four challenge types. Each challenge is characterized by a different task objective, available actions, and difficulty. The challenge type feature plays a pivotal role in interpreting other game interaction-related features accordingly. For example, suppose a student spent two minutes to finish a challenge. Two minutes may suggest poor performance in completing *Challenge A*, since it is a relatively easy task compared to other challenge types; in the meantime, two minutes may indicate good performance for *Challenge C,* because it often requires more actions and thus takes longer time to finish. We use a four-dimensional one-hot encoding feature vector to represent the four challenge types.

Second, the students' pre-knowledge feature is designed as a measure of students' prior knowledge about the relevant genetic concepts. Previous work has shown that students' pre-knowledge can serve as a significant predictive feature for student modeling [34]. Students with a better understanding of the concepts covered in the game are more likely to demonstrate higher performance on this feature. We use five questions extracted from the pre-test questionnaire, which are highly related to the genetic concepts in those challenges we focus on in this work. For each student, we use the ratio of correct answers as a predictive feature to represent students' initial knowledge.

Third, we design four game performance features associated with each challenge. Previous research suggested that demotivation and quitting are related to students' self-efficiency [22]. Since students' prior poor performance may negatively impact their self-efficacy related to genetic achievement, these poor performance could be predictive of future quitting behaviors. Therefore, students' in-game performance features could play an important role for modeling students' quitting behaviors. The four game performance features are formulated as follows:

- **Current challenge outcome:** This feature represents whether students quit the current challenge or not. We use a two-dimensional binary vector to represent it.
- **Time spent on the current challenge:** This feature captures the duration from when students started the challenge to when they leave it, where leaving is either quitting or finishing the challenge. Instead of using the absolute time, we use the Z-scores of the times calculated per challenge type, which enables the capture of challenge type-specific time information.
- **Ratio of wrong submission counts to total submission counts:** This feature measures students' overall performance on the challenge. In contrast to *Challenge A* and *Challenge B*, where students are allowed to make multiple submissions, in *Challenge C* and *Challenge D*, we consider the action of "putting an egg into a basket" as a submission (i.e., an implicit way of submitting an answer). We calculate the feature value based on the number of wrong egg-basket matchings divided by the total number of egg-basket matching attempts. It is worth noting that for those students who quit the challenge without making any submissions (i.e., 0 divided by 0 cases), we set the value of the feature to 1, which means maximum error rate.
- **Efficiency:** This feature is used to measure students' efficiency at completing the challenge. As mentioned in Section 3.1, there is a "goal move" for each challenge which indicates the minimum number of actions needed to successfully solve the challenge. For *Challenge A* and *Challenge B*, the minimum moves for a challenge is determined depending on the generated problem. For *Challenge C* and *Challenge D*, the minimum moves is fixed to 8, since 8 sorting actions are required per challenge. The efficiency feature is calculated as the ratio of the "goal action" number divided by the number of students' actions actually taken in cases where students completed a challenge (i.e., higher is better). On the other hand, when students quit a challenge, the negative of this ratio is used. This ensures that larger ratio values are better for this efficiency score, even for students who quit the challenge (i.e., giving a higher penalty to students who quit the challenge in earlier phase than students who quit in later phase).

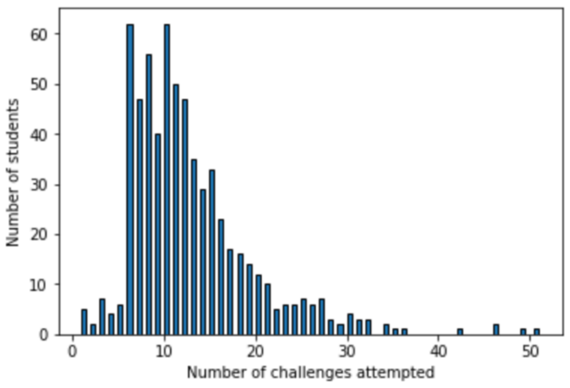
In addition to these six features, we create one additional explanatory variable representing the type of challenge the student will play next. It is hypothesized that students' quitting behaviors will not be homogeneous across all challenge types due to differences in the task objective, available actions, and difficulty. As a result, it is observed from Table 1 that the percentages of quitting class vary among challenges. In a runtime implementation of our framework, since the next challenge type is unobservable until the student chooses one, the predictive model can make inferences on quitting across all available challenge types, and then the framework can make scaffolding decisions based on probabilities of quitting across all challenge types. In our offline evaluation, our dataset allows us to get the next challenge type and use it as an additional explanatory variable. We investigate whether knowing the challenge type students will play next might serve as a strong predictor by exploring two different feature sets: one feature set enhanced with this next challenge information and the other feature set without considering it. This variable is represented as a four-dimensional one-hot encoded vector to represent the four types of challenge.

Moreover, to take advantage of students' historical gameplay information (Figure 2), we use *n*-gram models with a varying value of n. *N*-gram concatenates information from the observed sequence of *n* consecutive challenges (including the current and *n-1* previous challenges) students played as the final set of features for a prediction. As discussed, for the unigram model, we use the set of *M* features($F_c = \{f_c^1, f_c^2, \dots, f_c^M\}$) based on the current challenge F_{C_k} to predict student's challenge outcomes for the next challenge, L_{k+1}. For the bigram model, a variant of *n*-gram when *n*=2, 2*M* predictive features (F_{C_k} and $F_{C_{k-1}}$) are used from the two challenges (C_k and C_{k-1}) for predicting whether students will quit on C_{k+1} . The bigram-based concatenated features can be represented as follows:

$$F_{[C_{k-1},\, C_k]} = \{f_{C_{k-1}}^1, f_{C_{k-1}}^2, \dots, f_{C_{k-1}}^M, f_{C_k}^1, f_{C_k}^2, \dots, f_{C_k}^M\}$$

In more general, for the *n*-gram setting, *M*n* features are created based on the previous n challenges, which can be represented as $F_{[C_{k-n+1}, \dots, C_k]}$. If the observed challenges in the trajectory are less than n, we pad zeros for the features to represent the missing challenges. In this study, we explore *n* from 1 to 3 for the *n*-gram model to investigate if leveraging temporal information captured from a time-series challenge interaction is useful in predicting students' quitting behaviors.

Figure 2. Histogram of students' gameplay trajectory length



5. EXPERIMENTS

5.1 Machine Learning Models

We explore a suite of machine learning algorithms for our prediction task [33][17][25][20]. We implement our predictive models using six machine learning algorithms, including logistic regression (LR), decision tree (DT), naive Bayes (NB), support vector machine (SVM), random forest (RF), and feed-forward neural network (FFNN). To develop our models for the first five, we use Python 3.6 with scikit-learn, a Python machine learning library [26]. For the FFNN model, we use Keras with the Tensorflow backend [5].

Hyperparameter values for the first five machine learning algorithms are set to the default values specified by scikit-learn, which are briefly described below. For logistic regression, we used L2 norm regularization with a weight of 1.0, while the optimization method used is stochastic gradient descent. For Naive Bayes, we assume the likelihood of our features to follow a Gaussian distribution. For SVM models, we investigate linear SVMs with a radial basis function (RBF) kernel and a polynomial kernel. For the RBF-based SVMs, the regularization parameter *C* is set to be 1. The gamma changes along with the feature set used, which is calculated using 1 divided by the number of features. For example, the dimension of unigram BFS is 10, gamma is set to 0.1 in this case. Similarly, the dimension of bigram BFS is 20, thus gamma is set to 0.05. For the decision tree models, the scikit-learn library uses optimized version of the CART algorithm. The splitting criterion used is the GINI index. For the random forest models, we explored one hyperparameter to search the optimal number of trees to be generated in the model from {10, 50, 100}. Each decision tree in the random forest adopts the same settings used in the independent decision tree model. The number of features considered for splitting a node of a subtree is set to be *n*, where *n* is the number of all features in the original dataset. For example, for unigram BFS feature set, *n = 10* (The dimension of unigram BFS feature set is 10). The number of features considered for splitting in each tree is 3 ($\sqrt{10} \approx 3$). Neural network hyperparameters are often empirically determined. There are several categories of hyperparameters to consider, including optimization (e.g., optimizer, learning rate), model structure (e.g., the number of hidden units, initialized weights), and training criterion (e.g., regularization terms, loss function) [26]. When implementing the FFNN models, we adopted grid-search on structure-based hyperparameters, number of hidden layers and number of hidden units, which has significant influence on predictive performance. We tried the number of hidden layers from 1 to 5 and the number of hidden units in each hidden layer from {32, 64, 128}. For other hyperparameters, we utilize categorical cross entropy for the loss function and Adam stochastic optimizer [19]. We use the Glorot uniform initializer to generate initial weights [10]. The learning rate of training is set to 0.01 and dropout rate is set from {0.25, 0.5, 0.75}. We adopt a mini-batch gradient descent with the mini batch size of 128 when training. We trained all of these models on our two feature sets, feature sets without the next challenge information and feature set with the next challenge information under different n-gram settings. Another common model used for comparison is majority class-based method. In our dataset, the majority class is "completed", which accounts for 82.90%. This majority model predicts all students' outcomes for the next challenge as "completed".

We conduct student-level five-fold cross-validation to evaluate models' performance. In our problem, the quitting class is the category that we are most interested in, since our goal is to identify students at risk (i.e., students who might quit the challenge) and support their learning. Therefore, we evaluate the model performance with respect to recall rates of predicting the quitting class. Recall is calculated by *True Positive / (True Positive + False Negative)*, which refers to the percentage of the relevant class being correctly classified [16]. For algorithms for which we conducted grid-search on some hyperparameters, we chose those hyperparameters which helped algorithms achieved highest performance. For random forest model, the number of trees is set to 10. For FFNN model, we finally chose a 5-layer architecture with 128 hidden units with 0.25 dropout rate on each hidden layer.

Since the overall class distribution in our dataset is highly imbalanced, predictions by machine learning models trained with the dataset will likely be inclined towards the majority group [16]. In other words, models trained with the dataset would achieve a high predictive accuracy by predicting the majority class label (i.e., completed) for most of the data instances, but would suffer from inferring the incorrect labels for instances that belong to the minority class label (i.e., quit). To accurately recognize the 'quit' class, we conduct oversampling of the instances with the minor class label only for the training set, while the recall is measured for the intact, original test set. In the next section, we compare the results of models trained with the original dataset without oversampling and the dataset with oversampling applied.

5.2 Results

Table 2 shows the recall rates of the models induced with two variants of the feature set without conducting oversampling. As clarified in the previous section, we call the feature set which contains six features as Base Feature Set (BFS). As an alternative, we call the feature set that includes the next challenge feature (NC) type BFS + NC. Since the majority-class model classifies all instances into the "completed" class, the accuracy of this model is 83.90%. However, the recall rate for quitting of the majority-class model is 0.00%. A high accuracy value in our problem could not reflect model's ability to recognize quitting class, which is what we are interested in this work. Other models are able to correctly predict some quitting class, but the recall rates of quitting for most of the models explored in this work are not significant. It is not surprising because machine learning models are optimized to minimize the loss defined in an objective function and achieve a high predictive accuracy during training. In the experiment with the imbalance data, naive Bayes models show the highest recall rates outperforming other baseline models. The predictive accuracy of the best naive Bayes model is 73.63%, which is still less than the accuracy of the majority-class method. In addition, we find that the models' predictive performance for the quitting class does not generally improve when using *n*-gram representations ($n > 1$) which include students' historical gameplay information.

Table 3 reports the recall rates for predicting quitting of the next challenge, after we conducted oversampling on the dataset. It should be noted that all the experimental settings are identical, except that the results reported in Table 3 are obtained from a training set oversampled to have an equal distribution between the 'quit' class and the 'completed' class, while the same test set is used in the two experimental settings. Comparing without-oversampling to with-oversampling, nearly all models demonstrate significant improvements with respect to the recall rates of quitting class. Linear SVM models show an average of 57.30% improvement in the recall rates for all the conditions in pairwise comparisons. SVMs with the polynomial kernel and SVMs with the RBF kernel exhibit improvements of 60.23% and 60.64% on average, respectively. LRs, RFs and FFNNs demonstrate improvements on average of 12.65%, 48.84% and 50.47% respectively. The only exception is DT, which does not show improvements when utilizing bigram and trigram features.

Since most of the predictive models show improved performance after oversampling, further discussions are made based on the results of the models induced with the oversampled training dataset (Table 3). The best models for predicting quitting are SVMs with the polynomial kernel using the unigram feature set with the next challenge (NC) information. These models achieve a 76.91% in recall rate for quitting and an accuracy of 67.97%. SVMs with the RBF show the highest recall rates with bigram and trigram features. We compare the average recall rates for each feature set under the unigram, bigram and trigram settings. Overall, SVMs with the polynomial kernel achieves the highest recall rates (74.31% on average). SVMs with the RBF also show relatively high performance, which achieves 74.05%. Thus, we conclude that SVM models are the most robust machine learning method for our task. One distinguished advantage of SVM is the kernel tricks, which is a technique to project our original data into another feature space that offers enhanced capacity for models to classify data instances. Although deep feed-forward neural networks also learn salient features from the dataset through multi-level non-linear transformation, the models usually require a large amount of training data to successfully extract meaningful features. Our

Table 2. Predictive results of different models using different feature sets (Without oversampling)

		unigram	**2-gram**	**3-gram**	**Average**
Models	**Feature Set**	**Recall-quitting**	**Recall-quitting**	**Recall-quitting**	**Recall-quitting**
Majority-class	BFS	0.00%	0.00%	0.00%	0.00%
	BFS + NC	0.00%	0.00%	0.00%	0.00%
DT	BFS	27.47%	29.54%	28.58%	28.53%
	BFS + NC	33.12%	31.05%	29.94%	31.37%
LR	BFS	11.86%	12.50%	14.98%	13.08%
	BFS+NC	19.67%	16.80%	17.83%	18.10%
NB	BFS	49.84%	55.81%	**57.96%**	**54.54%**
	BFS + NC	**50.64%**	**56.61%**	56.21%	54.49%
RF	BFS	22.85%	24.44%	23.57%	23.62%
	BFS + NC	28.50%	26.99%	26.59%	27.36%
Linear SVM	BFS	0.00%	0.00%	0.00%	0.00%
	BFS + NC	0.00%	0.00%	0.00%	0.00%
SVM_poly	BFS	6.45%	6.21%	6.93%	6.53%
	BFS + NC	6.29%	5.10%	5.02%	5.47%
SVM _rbf	BFS	6.37%	5.65%	6.45%	6.16%
	BFS + NC	6.45%	5.02%	6.21%	5.89%
FFNN	BFS	12.10%	10.19%	15.45%	12.58%
	BFS + NC	11.86%	13.85%	14.49%	13.40%

Table 3. Predictive results of different models using different feature sets (Oversampling)

		unigram	2-gram	3-gram	Average
Models	**Feature Set**	**Recall-quitting**	**Recall-quitting**	**Recall-quitting**	**Recall-quitting**
Majority-class	BFS	0.00%	0.00%	0.00%	0.00%
	BFS + NC	0.00%	0.00%	0.00%	0.00%
DT	BFS	39.41%	27.31%	27.23%	31.34%
	BFS + NC	37.26%	30.81%	30.33%	32.96%
LR	BFS	60.03%	59.61%	60.51%	59.45%
	BFS+NC	70.86%	68.63%	70.22%	69.37%
NB	BFS	50.40%	59.39%	60.59%	56.39%
	BFS + NC	59.63%	60.19%	59.16%	60.00%
RF	BFS	42.44%	34.95%	35.67%	37.45%
	BFS + NC	42.44%	37.02%	38.30%	38.83%
Linear SVM	BFS	48.73%	49.76%	53.11%	49.42%
	BFS + NC	64.65%	65.45%	64.89%	65.18%
SVM_poly	BFS	62.02%	56.21%	57.09%	58.15%
	BFS + NC	**76.91%**	73.01%	69.51%	**74.31%**
SVM_rbf	BFS	61.94%	57.96%	60.91%	59.29%
	BFS + NC	75.96%	**73.09%**	**71.97%**	**74.05%**
FFNN	BFS	57.25%	59.71%	60.91%	58.89%
	BFS + NC	69.03%	67.52%	67.99%	68.02%

dataset is not large enough for deep learning models to effectively learn the intermediate features and achieve high performance.

Moreover, the results also support our hypothesis that the next challenge type feature improves model performance. These results support our hypothesis that students' quitting behaviors are highly influenced by the challenge type that they will choose next, rather than being generally predictable regardless of the challenge type. Moreover, we find that students' historical gameplay information obtained from the previous one or two challenges they interacted with does not seem to improve model performance, as demonstrated by models using the unigram features achieve the highest recall rates. The results suggest that students' historical gameplay information on previous challenges before the current induce more noise rather than adding more predictive power. This should be an interesting area that required further investigation in future work.

6. CONCLUSION AND FUTURE WORK

In this work we present a data-driven approach to modeling students' performance on challenges within an open-ended learning environment for genetics. We build an integrated model for all challenge levels, which can dynamically predict students' quitting behaviors on future challenges in their gameplay trajectory. In practice, the learning environment could use the predicted results from these models to decide on specific interventions to take. For instance, if the learning environment recognizes that a student is likely to quit at a challenge before starting, it could suggest another challenge to smooth their learning experience. To implement the predictive models, we engineered fine-grained features to describe student gameplay actions from their interaction log data and investigated the performance of different machine learning algorithms. The results show that SVM machine learning algorithm achieves the highest recall rate with respect to predicting students' quitting behaviors for our problem. We also find that using the next challenge information offers improved predictive capabilities for the models.

During our analysis of quitting behaviors in Geniventure, we identified two key situations. One type of quitting occurs immediately after students open a challenge, while the second occurs after extended struggle on the challenge. For the second type, it likely occurs because the difficulty of the challenge in the game does not match the students' current abilities. In this case, students' mastery of knowledge and their problem-solving skills in the game provide good evidence for predicting whether students will quit or not. However, for the first type, the reason for quitting is harder to ascertain. It could be caused by many factors that are not easily observable during the learning process. It could be a case that students are gaming the system or they might just want to take a look at challenge before deciding which challenge to play. The occurrence of this type of quitting behavior is less related to their in-game performance. Thus, differentiating these two types of quitting behaviors may help improve models' predictive performance and their abilities to support effective interventions. In the future, we may need to build models for more fine-grained types of quitting behaviors. Moreover, we may need to investigate more features that could reflect students' affective and cognitive states and dynamic progress of their mastery of content knowledge. In addition, we are also interested in investigating how students' learning gains are affected by interventions driven by our predictive models.

7. ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under Grant DRL-1503311. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] Asbell-Clarke, J., Rowe, E., Sylvan, E., and Baker, R. 2013. Working through impulse: assessment of emergent learning in a physics game. *Games+ Learning+ Society 9.0*.

[2] Baker, R. S. and Rossi, L. M. 2013. Assessing the disengaged behaviors of learners. *Design Recommendations for Intelligent Tutoring Systems. 1*, 153-164.

[3] Beck, J. E. and Gong, Y. 2013. Wheel-spinning: Students who fail to master a skill. In *Proceedings of the International Conference on Artificial Intelligence in Education*. Springer, pp. 431-440.

[4] Buffum, P. S., Frankosky, M., Boyer, K. E., Wiebe, E. N., Mott, B. W., and Lester, J. C. 2016. Collaboration and gender equity in game-based learning for middle school computer science. *Computing in Science & Engineering 18* (2), 18-28.

[5] Chollet, F. 2015. Keras. https://github.com/fchollet/keras

[6] D'Mello, S. K., Mills, C., Bixler, R., and Bosch, N. 2017. Zone out no more: Mitigating mind wandering during computerized reading. In *Proceedings of the 10th International Conference on Educational Data Mining*. pp. 8-15.

[7] Eck, R. Van. 2006. Digital game-based learning: It's not just the digital natives who are restless. *EDUCAUSE review 41* (2), 16-30.

[8] Fredricks, J. A., Blumenfeld, P. C., and Paris, A. H. 2004. School engagement: Potential of the concept, state of the evidence. *Review of Educational Research 74* (1), 59-109.

[9] Gee, J. P. 2003**.** *What video games have to teach us about learning and literacy. 1* (*1*), 1-4.

[10] Glorot, X. and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. pp. 249-256.

[11] Halawa, S., Greene, D., and Mitchell, J. 2014. Dropout prediction in MOOCs using learner activity features. In *Proceedings of the Second European MOOC Stakeholder Summit 37* (1), 58-65.

[12] Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., and Edwards, T. 2016. Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning. *Computers in Human Behavior 54*, 170-179.

[13] He, J., Bailey, J., Rubinstein, B. I., and Zhang, R. 2015. Identifying at-risk students in massive open online courses. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. pp. 1749-1755.

[14] Hicks, A., Peddycord, B., and Barnes, T. 2014. Building games to learn from their players: Generating hints in a serious game. In *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*. Springer, pp. 312-317.

[15] Hutt, S., Hardey, J., Bixler, R., Stewart, A., Risko, E. F., and D'Mello, S. 2017. Gaze-based detection of mind wandering during lecture viewing. In *Proceedings of the 10th International Conference on Educational Data Mining*. pp. 226-231.

[16] Jeni, L. A., Cohn, J. F., and Torre, F. D. La. 2013. Facing imbalanced data--recommendations for the use of performance metrics. In *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*. pp. 245-251.

[17] Karumbaiah, S., Baker, R. S., and Shute, V. 2018. Predicting quitting in students playing a learning game. In *Proceedings of the 11th International Conference on Educational Data Mining*. pp. 167-176.

[18] Kiili, K., Devlin, K., Perttula, T., Tuomi, P., and Lindstedt, A. 2015. Using video games to combine learning and assessment in mathematics education. *International Journal of Serious Games 2* (*4*), 37-55.

[19] Kingma, D. P. and Ba, J. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[20] Kloft, M., Stiehler, F., Zheng, Z., and Pinkwart, N. 2014. Predicting MOOC dropout over weeks using machine learning methods. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*. pp. 60-65.

[21] Li, Y., Fu, C., and Zhang, Y. 2017. When and who at risk? Call back at these critical points. In *Proceedings of the 10th International Conference on Educational Data Mining*. pp. 168-173.

[22] Margolis, H. and McCabe, P. P. 2004. Self-efficacy: A key to improving the motivation of struggling learners. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas 77* (*6*), 241-249.

[23] McElroy-Brown, K. and Reichsman, F. 2019. Genetics with Dragons: Using an online learning environment to help students achieve a multilevel understanding of genetics. Retrived from http://concord.org/.

[24] Mills, C., Bosch, N., Graesser, A., and D'Mello, S. 2014. To quit or not to quit: predicting future behavioral disengagement from reading patterns. In *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*. Springer, pp. 19-28.

[25] Min, W., Frankosky, M. H., Mott, B. W., Wiebe, E. N., Boyer, K. E., and Lester, J. C. 2017. Inducing stealth assessors from game interaction data. In *Proceedings of* the 9th *International Conference on Artificial Intelligence in Education*. Springer, pp. 212-223.

[26] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. *12*(Oct). pp. 2585-2830.

[27] Prensky, M. 2003. Digital game-based learning. *Computers in Entertainment (CIE) 1* (*1*), 21-21.

[28] Ritter, S., Anderson, J. R., Koedinger, K. R., and Corbett, A. 2007. Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review 14* (*2*), 249-255.

[29] Rowe, J., Mott, B., McQuiggan, S., Robison, J., Lee, S., and Lester, J. 2009. Crystal island: A narrative-centered learning environment for eighth grade microbiology. In *Workshop on Intelligent Educational Games at the 14th International Conference on Artificial Intelligence in Education.* pp. 11-20.

[30] Rowe, J. P., Shores, L. R., Mott, B. W., and Lester, J. C. 2011. Integrating learning, problem solving, and engagement in narrative-centered learning environments.

International Journal of Artificial Intelligence in Education 21 (*1-2*), 115-133.

[31] Shute, V. J., and Ke, F. 2012. Games, learning, and assessment. *Assessment in game-based learning*. Springer, pp. 43-58.

[32] States, N. L. 2013. *Next Generation Science Standards.* Washington.

[33] Taylor, C., Veeramachaneni, K., and Reilly, U. O. 2014. Likely to stop? Predicting stopout in Massive Open Online Course. *arXiv preprint arXiv:1408.3382*

[34] Wan, H., and Beck, J. B. 2015. Considering the influence of prerequisite performance on wheel spinning. In *Proceedings of the 8th International Conference on Educational Data Mining.* pp. 125-139.

[35] Wang, P., Rowe, J., Min, W., Mott, B., and Lester, J. 2017. Simulating player behavior for Data-Driven interactive narrative personalization. In *Proceedings of* the *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*. pp. 255-261.