Fueling Prediction of Player Decisions: Foundations of Feature Engineering for Optimized Behavior Modeling in Serious Games

V. Elizabeth Owen[1] · Ryan S. Baker[2]

Abstract
As a digital learning medium, serious games can be powerful, immersive educational vehicles and provide large data streams for understanding player behavior. Educational data mining and learning analytics can effectively leverage big data in this context to heighten insight into student trajectories and behavior profiles. In application of these methods, distilling event-stream data down to a set of salient features for analysis (i.e. feature engineering) is a vital element of robust modeling. This paper presents a process for systematic game-based feature engineering to optimize insight into player behavior: the IDEFA framework (Integrated Design of Event-stream Features for Analysis). IDEFA aligns game design and data collection for high-resolution feature engineering, honed through critical, iterative interplay with analysis. Building on recent research in game-based data mining, we empirically investigate IDEFA application in serious games. Results show that behavioral models which used a full feature set produced more meaningful results than those with no feature engineering, with greater insight into impactful learning interactions, and play trajectories characterizing groups of players. This discovery of emergent player behavior is fueled by the data framework, resultant base data-stream, and rigorous feature creation process put forward in IDEFA—integrating iterative design, feature engineering, and analysis for optimal insight into serious play.

Keywords
feature selection; feature engineering; educational data mining; behavior modeling; game-based learning; serious games

Acknowledgements
This work was made possible by a grant from the National Science Foundation (DRL-1119383), although the views expressed herein are those of the authors' and do not necessarily represent the funding agency. We would also like to thank Richard Halverson, Constance Steinkuehler, Kurt Squire, and Matthew Berland.

[1] V. E. Owen (corresponding author)
University of Wisconsin-Madison
v.elizabeth.owen@gmail.com

[2] R. S. Baker
University of Pennsylvania, Graduate School of Education
rybaker@upenn.edu

Fueling Prediction of Player Decisions: Foundations of Feature Engineering for Optimized Behavior Modeling in Serious Games

Abstract

As a digital learning medium, serious games can be powerful, immersive educational vehicles and provide large data streams for understanding player behavior. Educational data mining and learning analytics can effectively leverage big data in this context to heighten insight into student trajectories and behavior profiles. In application of these methods, distilling event-stream data down to a set of salient features for analysis (i.e. feature engineering) is a vital element of robust modeling. This paper presents a process for systematic game-based feature engineering to optimize insight into player behavior: the IDEFA framework (Integrated Design of Event-stream Features for Analysis). IDEFA aligns game design and data collection for high-resolution feature engineering, honed through critical, iterative interplay with analysis. Building on recent research in game-based data mining, we empirically investigate IDEFA application in serious games. Results show that behavioral models which used a full feature set produced more meaningful results than those with no feature engineering, with greater insight into impactful learning interactions, and play trajectories characterizing groups of players. This discovery of emergent player behavior is fueled by the data framework, resultant base data-stream, and rigorous feature creation process put forward in IDEFA—integrating iterative design, feature engineering, and analysis for optimal insight into serious play.

Keywords

feature selection; feature engineering; educational data mining; behavior modeling; game-based learning; serious games

1 Introduction

In learning and making meaning in our digital age (Steinkuehler, Barab & Squire, 2012), serious games can be powerful, immersive educational vehicles (Gee, 2003) and provide large data streams for understanding player decision making and growth (e.g. Shute, 2011; Clark et al., 2012; Loh, 2012). These data-rich, immersive learning environments are thus prime candidates for application of educational data mining (EDM) and learning analytics (Baker & Siemens, 2014), which leverage big data in educational contexts to optimize insight into player decisions and growth trajectories. However, event-stream analysis models can be limited by the quality of data features curated for input—and so, to optimize the power of educational data mining for insight into player decision and profile modeling, rigorous approaches to data collection, cleaning, and feature engineering are critical (Baker, 2013). Indeed, distilling event-stream data down to a set of salient features for analysis (i.e. feature engineering and feature selection) is a critical element of robust modeling in data mining (e.g. Guyon & Elisseeff, 2003; Fogarty, 2006; Sao Pedro et al., 2012). Robust data mining models are resilient to minor forms of noise and bias (e.g. Fogarty et al.,2004). Models can be made more robust through using sufficient data, appropriate algorithms, and building models based on a valid and appropriate set of data features (see section 3.2.2; e.g. Sao Pedro et al., 2012).

This article discusses key data collection and feature engineering principles specific to serious games, vital for optimizing insight into patterns of user interaction (and thus player profile modeling) within the game space. This has been an especially vital component in applications involving serious games (e.g. Malkiewich et al., 2016; Owen et al., 2016), since analysis in this medium often targets a complex blend of player engagement *and* growth around specific learning goals (cf. Ifenthaler et al., 2014). Aligned game design and data collection enables high-resolution feature engineering, which can be honed through critical, iterative interplay with analysis. These key components of a strong feature foundation can be broken down into several layers for optimization of insight into player behavior and player profiling (Fig. 1).

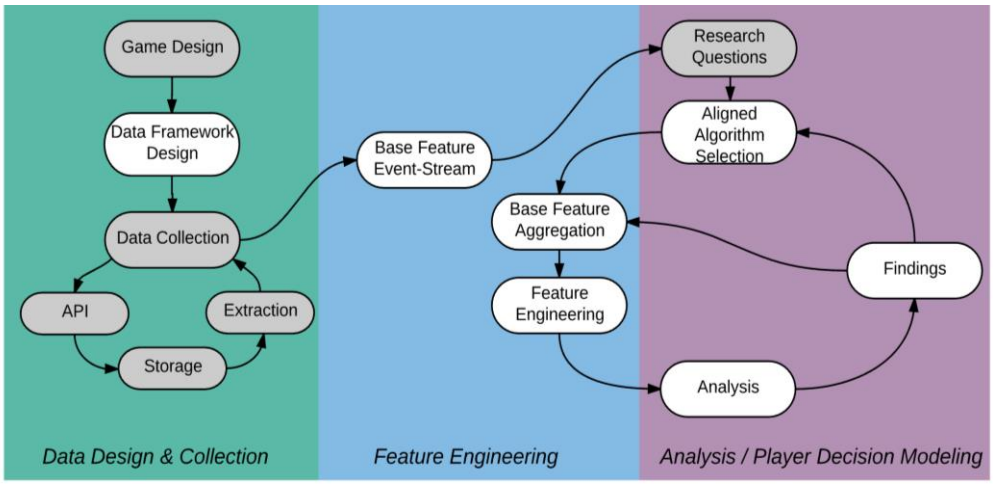

Fig. 1 A process of data design, feature engineering, and analysis: the IDEFA framework.

This process for optimizing feature engineering for insight into user behavior is presented in this article as the IDEFA framework (Integrated Design of Event-stream Features for Analysis). The

framework's highlighted (white) areas show major components of a strong data system, interwoven with iterative analysis in game-based data mining. Key elements include a strong data collection schema, which provides well-organized and interpretable data (clearly corresponding with the designed mechanics of the game), that is also comprehensive (capturing the full interplay and richness of student interaction and system feedback). This kind of strong data framework specific to games (such as ADAGE, e.g. Halverson & Owen, 2014) provides a substantial foundation for the next phases: base feature aggregation and feature engineering. As the ecology shows, aggregation at the unit of analysis relies strongly on the algorithm chosen for a particular investigation. Once event-stream data is aggregated at the unit of analysis (such as the student, or a phase of gameplay), then additional features can be created using a blend of human judgement and systematic application of mathematical operations. For example, we may have aggregated data for each student about time spent playing the game (among a wealth of other information). This would be a base feature from the core data collected from play. Moving to feature engineering, it's then possible to create *new* features (not present in the base data stream) by applying a computational lens—for example, the *standard deviation of time* spent on each level of the game, created by taking the base feature of time and applying a mathematical function. This kind of feature engineering can be done across a myriad of base features to create a new set of data that incorporates systematic computation across base features with the curation of human judgement. These new features can provide fine-grained information about student interaction and behavior, creating nuanced variables that can sharpen learner insights in final analysis. Based on the results of the next step—plugging this rich data into analysis—feature engineering is conducted in an iterative fashion, where features are built and their impact on the modeling process is studied. In this way, organic patterns in the data are allowed to emerge through systematic, iterative feature engineering and analysis. These emergent patterns provide critical insight into player trajectories in serious games, a medium in which exploration and boundary-testing are inherent norms (Squire, 2011; Salen & Zimmerman, 2004).

Implementation of this process in serious games supports the modeling of player decisions within diverse, mined gameplay pathways. Building on the body of research which utilizes feature engineering for data mining in game contexts (c.f. DiCerbo & Kidwai, 2013; Baker & Clarke-Midura, 2013), we empirically investigate specific approaches to feature engineering for strong insights into learner behavior in serious games. In particular, we investigate how the presence or absence of rigorous feature engineering impacts the performance of a model predicting productive player exploration within a serious game. This feature engineering applies human judgement paired with mathematical functions systematically applied across features (cf. Paquette et al., 2014). The contrast of these models offers empirical insight into the emergence of nuanced behavioral patterns. These insights into player decisions and profiles of play trajectories are fueled by the data framework, resultant base data-stream, and rigorous feature engineering foundation put forward in this approach to data design and feature engineering for optimal insight into serious play.

The following paper will discuss the IDEFA framework in two parts, with an extended example from recent research in each segment. Part I, data design and collection (green section, Fig. 1), is intertwined with game development and sets the data foundation for subsequent strong feature engineering. Part II focuses on a process of feature engineering itself, discussing the creation of optimal event-stream features through interplay with the raw data stream, methodical feature engineering lenses, and iterative analysis. Finally, we empirically investigate the application of the complete IDEFA feature engineering process through the comparison of analysis models, contrasting behavior insights with input variables generated with and without use of the feature engineering system presented in this paper.

2 Feature Engineering Roots: Data Framework Design and Data Collection (IDEFA Part I)

Any process of making meaning out of data, whether involving thorough feature engineering or more bottom-up processes, is dependent on the integrity, quality, and scope of the original data. This section details elements of this data foundation, as represented in the initial stages of the feature engineering process (left panel, Fig. 1).

A well-designed data framework is built to produce comprehensive, consistently labeled data aligned with learning design (which sets a strong foundation for accurate, detailed features for robust analysis and nuanced behavior insights). Ideally, a well-designed game will have game events that can be interpreted directly in terms of the types of competencies and learning that the designer wants to measure (e.g. Shute & Kim, 2014); the process of this design produces data that can also be interpreted in terms of the engineering of other features. This also allows for interpretable data models, which can directly inform data-driven design of the game itself – i.e., when data is interpretable and design-aligned, outcomes of analysis can be more easily translated to direct feedback into design. For example, if sequence mining results show player attrition happens largely at a given point in the game, and that point is clearly identified in the data as a specific quest or progress checkpoint, then design can be adjusted to support players through the chokepoint. This support can take the form of EDM-fueled automated feedback to learners (e.g. Berland et al., 2014), also enabled by clear, design-aligned event-stream data.

Thus, a strong data framework (when accurately implemented into the game using an aligned API), is a thorough, clearly labeled data stream of player interactions and game events. This comprehensive, moment-by-moment log file data then can support many investigations into player behavior in iterative data mining analysis—a data foundation that is especially important when the game (or suite of games) is live at scale on the consumer market, and data logging updates are not easily pushed without repercussions for the whole system.

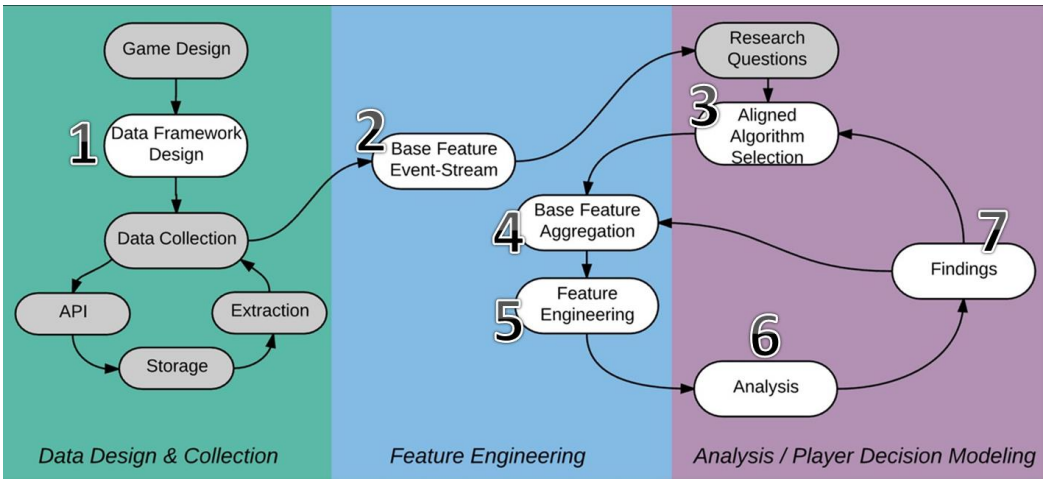

Fig. 2 Numbered elements in the IDEFA ecology.

This section discusses details of this foundation: data framework design, and the resulting event-stream data (elements 1 & 2 in feature engineering for optimal data mining process, Fig. 2). The aligned real-world system components of the physical game's design and the back-end data system will be

touched on briefly, then an example given of this foundational process flow (through element 2, Fig. 2) based in the biology game *Progenitor X*.

2.1 Data Framework Design

Within recent efforts in structuring serious game data (e.g. Chung, 2015; Owen et al., 2013), a strong framework for game data capture provides several key functions: comprehensive data, clear organization, and consistency in labeled events throughout the game. In practice, this can take the form of several core types of recorded events. Specifically, *player actions* and *system feedback* are captured as part of a high-resolution data framework (e.g. Danielak, 2014); additionally, it effectively marks player *progression* (e.g. the user's current level, quest, or objective) within the scope of the full game. Each core event type is then associated with important *context* (e.g. timestamp and player ID) as well as the *result* of the player action (e.g. win/lose, score, or more fine-grained outcomes) when appropriate (cf. Serrano-Laguna et al., 2017; Hao et al., 2016). Details of each category are discussed below.

Player actions and system feedback are core events for recording, because they represent the player and the system interacting with one another. Player actions simply capture input actions the player makes with the system. Traditionally, this can include clicks or tap the user has with the interface; if the game has more sophisticated user input like voice recognition, eye-tracking input, or body movement (e.g. in virtual reality games), these can be captured as well. At a deeper level, player action also constitutes engagement with core verbs of play aligned with learning. For example, in some contexts it is more meaningful to capture the core semantic behaviors rather than every click or tap the player makes (i.e. capturing mathematically meaningful player actions such as attacking a "17" skeleton with a "3" weapon rather than all movements around the environment in *Zombie Division*—(Habgood & Ainsworth, 2011). Generally, player actions constitute users interfacing (or communicating) with the game.

The converse, or the game communicating with users, can be described as system feedback. This describes feedback events the game shows to players that help move them forward, including instructional cues to the learner (like a tutorial popup), organic consequences of actions (e.g. the game decreasing an avatar's health if she tumbles down a hill), or narrative vehicles (e.g. a cut scene the player might earn as a result of reaching a game milestone). In complex system games such as SimCity, where changes in the city may be the result of multiple player actions over time, this feedback may take the more complicated form of a state change (e.g. change in city-wide pollution level) from turn to turn or at a given increment of time. Generally, these feedback events show what students are seeing from the system, which can make their next actions easier to interpret.

Play progression marks the start and end of key stages of the game as defined in the game design, so that a players' current progress in the game can be seen in relationship to any given action or system event. If games are a designed experience (Squire, 2006), these events help us understand where the player is within this experience at any point in time. This marking of player milestones conveys how far the learner is into the game (e.g. which level, phase, or progress point the user is working within at any given point in the game), as well as capturing valuable context about their behavior. These progress markers in play can vary widely based on game genre: e.g. game objectives, phases, or difficulty levels. For example, puzzle games like *Tetris* and *Candy Crush* have difficulty levels people play, finish and move on to the next; role-playing games like *World of Warcraft* might have multiple quests or objectives users finish to progress. In the case of *Candy Crush*, each level start and end would be recorded in the

data, along with the name of the level (e.g. "Level 1") for clear identification[1]. Clearly recording this user progression in the data gives critical context to player actions. For example, if we know that many players are quitting at the same point in the game, but we're not recording the game level in which this happens, it's a major loss of insight. Knowing where the player is in the designed experience of the game enables better interpretation of player actions and system feedback.

For the core events of player action, system feedback, and progression markers, additional associated fields of context and result are among them (e.g. xAPI; Kevan & Ryan, 2016). Context attached to each core event includes basic information about who, when and where—such as player ID, timestamp, and where on the screen the player clicked (x,y coordinate) during a player action. Result information shows an in-game evaluation of a player action or progression marker—for example, at the end of a quest, level, or game challenge, does a player win or lose? Finer-grained outcomes can be captured as well—such as types of success or failure (e.g. in the case of *Progenitor*, either cycle failure by cell death, or cycle failure by wrong cell collection), the player's score (e.g. in points or a percent), or categorical evaluation (e.g. addition by use of grouping strategy, addition by use of carrying strategy, etc.).

Overall, player actions, system feedback, and progression markers (enriched with relevant context and result information) support a comprehensive, clearly defined, and consistent game data framework.

2.2 Data Framework: Interplay with Game Design, Backend Data Systems, and Base Event-Stream

Thinking though these core data framework events (e.g. player action, system feedback, and progression markers) *during* the game development process can greatly support learning game design. Since in good games progression and mechanics are intrinsically aligned to targeted learning objectives (Gee, 2005), thinking through learning mechanics as producers of performance data up front can enhance design by: 1) better aligning learning objectives to game mechanics, and 2) helping tune game mechanics to produce clearer performance data. This principle is present in cognitive tutor design, for example, in which each tutoring problem is aligned with learning content (cf. Koedinger et al., 2012), and produces clear evidence of performance at the end of each question. This principle is reflected as well in approaches like Evidence Centered Design (ECD; Mislevy & Haertel, 2006), in which designed tasks are clearly aligned with learning evidence. However, unlike ECD, our emphasis on designed measures of performance does not limit the full range of data collected or the approach to analysis—it simply helps initial design thinking around learning objectives and game mechanics.

Although ideally intertwined with game design, data collection is logistically tied to back-end systems by definition (both grey elements, Fig. 2). To implement the data framework into game code itself, a standard logging library (or API) can be created to reflect the data framework, and used to create data "hooks" in the base code of the game. Moving to the next grey element under "data collection" (Fig. 2), the data must reside in a database (or several) and stored in a form that facilitates extraction (e.g. export in the form of a csv [comma separated values document] or JSON file).[2]

---

[1] It's worth noting that this is useful across game genres. For example, in non-linear and open-world games, multiple kinds play progression may also be recorded; e.g. in *Skyrim* (an open world role playing game, where players are not on a linear track), multiple opened quests can be tracked at once, as well as the player's current "xp" or experience level (points gained as the player does more and more in the game).

[2] Other ETL (extract, transform, load) processes may be involved to convert the data into a final legible form that reflects the data framework design.

Thus, a "base feature event stream" (element 2, Fig. 2) is generated, created by the framework-guided API hooks when students interact with a game, and then stored in a database, and extracted in a form a researcher can use. The base feature event stream is referred to here as the event-by-event log of interactions that occur when users play an instrumented game. This forms the "raw" data, because it reflects the actual data hooks that are embedded in the game data (with minimal processing or manipulation). The events coming in with the raw data stream can be called our "base features", since they give us the foundation of events with which we make variables for analysis.

2.3 Example Design, Data Framework, and Event Stream: *Progenitor X*

By looking at the application to an existing game, we can illustrate the data principles above in concrete detail. We'll take the example of *Progenitor X*, a biology learning game, created at a game research and development lab at a major Midwestern university[3]. First, we'll look at *Progenitor* design, interplay with the data log framework design, and the resulting event-stream data.

2.3.1 Game Design: Progenitor X

Progenitor X is based in a biology content model, focused on regenerative medicine and stem-cell science. Set in an apocalyptic world overrun by ravenous zombies, *Progenitor* positions the player as a regenerative biologist with the power to save the human population by curing zombie-infected patients. The biology content model is manifested in the main goal of the game: to cultivate healthy cells and assemble new tissues in order to repair organs contaminated with the zombie infection. *Progenitor* is designed to teach cutting-edge knowledge and processes in stem-cell science, rooted in serious collaboration with top regenerative medicine scientists at the Wisconsin Institute for Discovery (WID).

Fig. 3 The *Progenitor X* virtual lab grid.

Progenitor gameplay centers around a virtual mobile lab (Fig. 3) that students use to engage in practices of regenerative medicine. A process derived from professional practice provides a simplistic but

[3] source generalized for purposes of blinding the manuscript

coherent account of real scientific procedures, designed for accessibility to the study demographic of secondary school students. The game invites students to engineer healthy tissue and organs through the cultivation and differentiation of adult epithelial stem cells (Owen et al., 2013). Key virtual lab procedures include a process of *populating* cells, *treating* the cells, and *collecting* the newly transformed cells. Beginning at the cell level, students *populate* the lab grid with healthy surface-layer skin cells *(populate/start)*. Next, players *treat* the cells with a special process to transform them into a new kind of cell. In the final phase, students c*ollect* the new cells after cultivating them to increase in number.

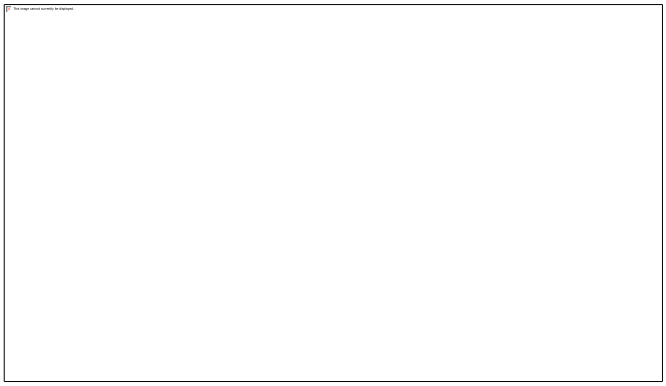

Fig. 4 Sequence of core content mechanics in *Progenitor X*.

This series of actions—start, treat, and collect—comprises a core cycle of play in *Progenitor*. These start/treat/collect cycles are the building blocks of the game, and occur in increasing difficulty in three systems-level biology layers: the cell level, the tissue level, and the organ level. Learners experience these core cycles in each nested phase. First, they cultivate cells, which then become the building blocks of healthy tissue, which in turn is used to repair organs, the largest system component. Throughout these cell, tissue and organ phases, the cycles build with progressively complex cell types and treatment procedures. Together, these start/treat/collect cycles at the cell, tissue and organ level provide a multi-layer experience with regenerative biology in an engaging narrative context.

2.3.2 Data Framework Design and Collection: Progenitor

The data captured in *Progenitor X* was aligned to these core learning mechanics and progress markers in play, utilizing the data framework ADAGE[4], the "Assessment Data Aggregator for Game Environments" (Halverson & Owen, 2014). Fundamentally, ADAGE is an event-stream data framework designed to capture salient learning data in educational games.

ADAGE captures core event types analogous to those described in the prior data framework section: Player Action, System Feedback (also called System Events), and Progress Markers (also called progress "Units"). Progress Units mark basic progression through the game, based in the learning design detailed above. Specifically, in *Progenitor X*, progress through the learning space is divided into objectives. There are eight objectives total in the game, with the eighth one being the final difficult level (or "boss" level). Each objective, in turn, is made up of cycles (Fig. 5). A cycle represents the core

[4] www.adageapi.org

learning unit. (This cycle, or base unit, is where students transform normal cells into new tissue growth, through the mechanics of start/treat/collect.)

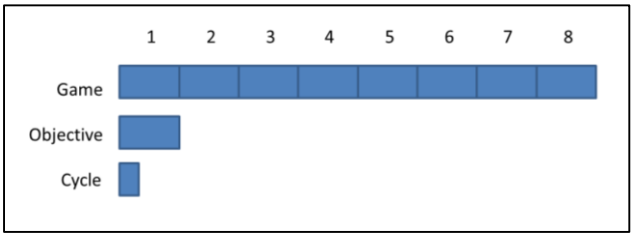

Fig. 5 A representation of nested progress mechanics (units) in *Progenitor X*.

An objective represents the largest repeating chunk of play progress, while the cycle represents the smallest. On the cycle level (using start/treat/collect actions), students transform skin cells through stem-cell science into new tissue and healthy organs. Thus, the cycle is a key learning unit, where *Progenitor's* biology content model aligns with virtual laboratory procedures in the game. These units provide a baseline for understanding learner navigation through the game. In turn, they provide a basis for capturing event-stream student interaction with designed learning mechanics.

Thus, progress Units in *Progenitor* were represented at three different levels: game start/end, objective start/end, and cycle start/end. Within each of these, Player Actions and System Events were identified. Each of these three core data types were linked with context and a result (when appropriate). Fig. 6 gives a simplified view of the data framework structure with some example values. It's worth noting that ADAGE represents a specific schema of the data. However, its main components (integrated into a game-specific framework) synthesize broad structural elements (progress markers, player actions, event context and results) distributed across recent efforts in learning data collection—including CRESST data standard development (e.g. Chung, 2015), GlassLab[5] evidence-based game design (cf. Corrigan et al., 2015), and ETS log file exploration (Hao et al., 2016).

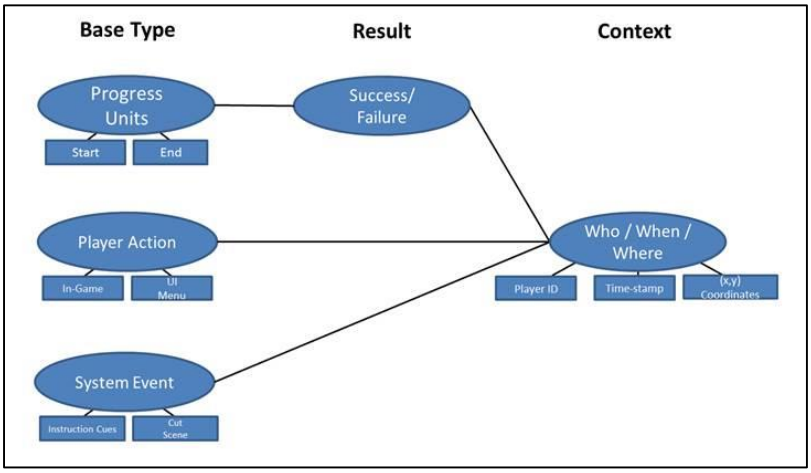

Fig. 6 A simplified representation of core data types, result, and context fields in ADAGE.

[5] https://www.glasslabgames.org

Specific to *Progenitor*, the following kinds of data were collected:

- Player Actions included all taps and drags, UI button usage (including the "back" and "forward" buttons for reviewing directions, an almanac button accessing biology terms, sound settings, the cell/tissue/organ lab view tabs, and the "quit" button) as well as laboratory tool use described above (start/treat/collect).
- System Events included tutorial pop-ups, screen view changes (e.g. moving from the cell lab view to the tissue lab view), and presentation of cut scenes to players.
- Progress Units, as mentioned above, included game, objective, and cycle starts/ends. Objective starts/ends were labeled with the corresponding objective's name (e.g. "Objective 1"), and cycle starts/ends were labeled with the type of cycle (cell/tissue/organ) and linked with the corresponding objective.
- Context, attached to each of the three core event types above, provided information about who, where, and when for an event (including player ID, session ID, timestamp, location of player, screen coordinate of interaction, device type used, OS version, game version, etc.).
- Result was a field that was often connected to the Player Actions within a cycle (start/treat/collect) and to the end of a progress unit like a cycle or objective. It was populated with varying data types as needed (e.g. a categorical success/fail, or a score such as a point total or percent).

This data thus represents the base event stream of "raw" features, supported by the data framework elements of Player Actions, System Events, and Units (enhanced with context and result, as appropriate). Implemented with a Unity (game client) API, the event-stream data was stored in a Mongo DB database. In the feature engineering section to follow, these "raw" event stream features are discussed in greater detail as they are aggregated for analysis in element 4 of the larger feature selection process (see Table 4).

These foundational steps for a strong data stream support feature engineering. The next section discusses Part II of the IDEFA process (Feature Engineering and Analysis, Fig. 2) in subsequent phases: feature aggregation, feature engineering, and iterative interplay with analysis.

3 IDEFA Part II: Feature Engineering for Optimal Analysis Insight

After a foundational base data stream is established in conjunction with game design and backend systems, this section discusses feature engineering with this data for optimizing iterative analysis (elements 3-7, Fig. 7). Starting with research questions and algorithm selection, it then explores cyclical feature aggregation and engineering for analysis. Overall, this systematic approach (Fig. 7) to data transformations for feature engineering help optimize input data for analysis. In addition, the iterative nature of the process allows for data-driven exploration of new features to be engineered, using analysis results to illuminate key interaction data of interest. Thus, once a final analysis model is produced, it can surface nuanced game interactions that help us gain insight into player behavior. In the following paragraphs, the elements of this process will be described in detail, and the discussion of *Progenitor X* is extended to give a concrete example.

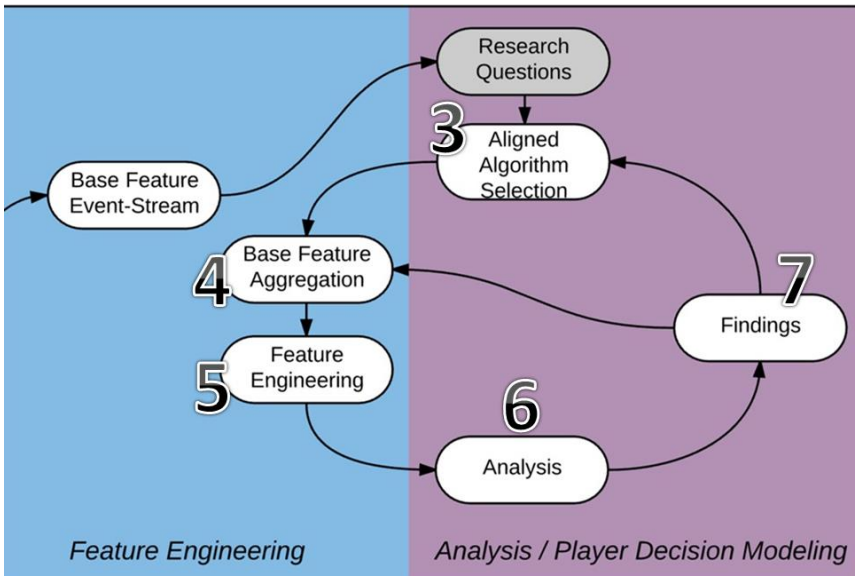

Fig. 7 A zoomed view: feature engineering and analysis in the IDEFA framework.

3.1 Algorithm Selection and Research Questions

As a starting point, analyses are generally created in response to a desired insight, or a guiding research question, although there are relatively more and less bottom-up approaches to data mining. The initial research question, or research questions, are represented in the diagram in grey (top middle section, Fig. 7), in which one or more research questions set guiding constraints which are used to select an appropriate algorithm (element 3). For example, in a quest-driven math game, one might want to understand if students' learning performance in one kind of skill (e.g. the counting 1-5 quest) is connected to learning performance in another kind of skill (e.g. the numeral recognition quest). Put simply, are performance on math quests related to one another? (We will call this RQ.)

For each research question asked, there are clear implications for practical analysis processes (white elements in Fig. 7). To be clear, the ecology map elements 3-6 show a process for a *single* given analysis relative to a question. However, for any given research question, there may be many different analyses that can be done. For each question, it's helpful to pick one overall method at a time that can shed insight, then follow the trail of findings from each one that may open up new analyses to be conducted or features to be engineered. In this way, iterative analyses can help uncover the shape of the data one by one (somewhat similar to clearing the Fog of War in *Civilization V*, which is a cloud cover obscuring unexplored parts of the map, by sending out one voyager at a time to uncover new terrain).

In choosing a method to start, we get to element three on the chart: algorithm selection. A selected method can be anything in a range from simple visualization to advanced machine learning techniques. For simplicity in our first example, we'll select an algorithm that can help us understand the relationship between quest scores (in alignment with RQ above): correlation mining (Baker, 2010), the practice of computing correlations (in a principled fashion) between multiple predictor features and one or more predicted features.

3.2 Feature Selection, Aggregation, and Engineering for Analysis

3.2.1 Base Feature Aggregation

After an algorithm is selected, we need to figure out what data we are using in the analysis. In "base feature aggregation" (element 4, Fig. 7), there can be two phases. First is base feature selection, in which we literally pick variables (or "features") from the base event-stream data (element 2) that are thought to be the most salient for analysis. Secondly, those features are aggregated at the unit of analysis.

For example, in correlating quest scores for RQ above, we know we need information on quest name and score. In the raw or "base" data, these might show up as *quest_name* and *quest_score*. To make sure only quests show up that students completed for a final score, a *quest_complete* event (a "progress marker" in the data framework discussed earlier) would be important. If we are analyzing at the student level, e.g. correlating scores for a single student at a time, then we will also need *student_ID* from the base data. So, in base feature selection (step 1), we have chosen these features: *student_ID*, *quest_complete*, *quest_name*, and *quest_score*. If these features are pulled from the base stream, they may now look like this (shown in table form for ease of viewing):

Table 1

Base Feature Selection Example

student_ID	quest_complete	quest_name	quest_score
A1	TRUE	Number_recognition_1_to_10	340
A1	TRUE	Counting_1_to_5	200
A1	TRUE	Counting_6_to_10	62
A2	TRUE	Counting_1_to_5	781
A2	TRUE	Number_recognition_1_to_10	635
A2	TRUE	Counting_6_to_10	433
A3	TRUE	Counting_1_to_5	848
A3	TRUE	Number_recognition	601
A3	TRUE	Counting_6_to_10	592
…	…	…	…

Now, we have information on only finished quests, complete with quest score and student ID. However, we want to analyze the correlations between our features at the student level (e.g. have one pair of values for each student, with each student having no more than one row of data). The data above, as it was pulled from the base data stream, is currently structured by quest completed, and each student can have multiple quests (and thus multiple rows) of data. Thus, aggregation (step 2) is next (i.e. condensing the data to the unit of analysis)—in this case, getting the data summarized per student. After aggregation, the final feature set for correlation mining might look like Table 2 below. However, note that this is a simplified example; typically, we would develop many more features in order to leverage large quantities of data. With enough data it is possible to explore multiple hypotheses at once rapidly; as Alan Newell once stated, "you can’t play twenty questions with nature and win" (Newell, 1973).

Table 2

Feature Aggregation Example

student_ID	number_rec_quest_score	counting_1_5_quest_score	counting_6_10_quest_score
A1	340	200	62
A2	635	781	433
A3	601	848	592
…	…	…	…

In this form, the selected features have been aggregated at the unit of analysis (the student), and one step closer to being ready for correlation. This is a result of feature selection from the base data stream *and* data aggregation, two steps in "base feature aggregation" (element 4, Fig. 7). (It is worth noting that while feature selection and aggregation are often two discrete steps in thought process, they can be executed in one chunk of code for implementation. The sample tables of selected features vs engineered features are shown simply for the sake of illustration. The details of how selection and aggregation are executed may vary by system—for example, ETL (extract, transform, load) practices differ according to the kind of database used (e.g. schemaless or relational).) In more complex analyses, the aggregation step may also include merging of event-stream features and external data sources, such as survey results, a posttest, or post-system outcomes.

3.2.2 Feature Engineering

After base data aggregation, the next process in our ecology is feature engineering (element 5) using mathematical operators. Feature engineering is the creation of new variables from an existing data set (e.g. Baker, 2013). In contrast, feature selection can be described as picking features from an existing set of variables or data stream, while feature engineering creates *new* ones. The two processes are compatible. While there is some mild feature selection in considering how to engineer features, full feature selection often takes place after feature engineering. While feature engineering remains more of an art than a true engineering discipline, there are several principles for the effective design of features that can be considered. First, a good feature has *construct validity* in terms of being representative of the construct that is desired to be represented. For instance, if one is trying to represent systematic guessing, both fast actions and similar answers given consecutively may be good representations of that behavior. Using features that domain experts recognize as having good construct validity has been found to lead to more generalizable models (Sao Pedro et al., 2012). Second, a good feature has *predictive validity* in terms of actually correlating to the construct of interest – i.e. a good feature leads to good models. Third, a good **set** of features has low overlap and high *coverage* (also thought of as *content validity)*, both conceptually as well as in terms of statistical collinearity. Rather than engineering a large number of very similar features, it is beneficial to engineer features that are different from one another and cover the full space of the content being represented. This set of features can be developed in several ways – two successful approaches include structured brainstorming and winnowing of the resultant feature set according to multiple criteria (e.g. Baker et al., 2011) and interviewing a domain expert, representing their process as a model, and then extracting the operators they reason with (e.g. Paquette et al., 2014).

Processes can be very different in each; for example, take the RQ features aggregated above (Table 2). Let's assume we want to be able to look at counting scores vs. number recognition scores in a

single correlation. In order to do this, the three values per student would need to be transformed into two values per student—one for number recognition, and one for counting. Counting currently has two values (one score for 1-5 and the other for 6-10), and one way to use these to create a single new variable is to take the average of the two. Alternatively, one could look for the variation across scores by taking a standard deviation, or one could set a cut-off across the two variables and check whether either is greater than that cut-off. The number of potential variables that can be created is limited only by the creativity of the researcher, although the process is generally more effective when guided by theoretical understanding (e.g. Sao Pedro et al., 2012). This is feature engineering: creating new variables from one or more previous variables. By feature engineering these variables, the data is then ready for input into a data mining analysis. The resulting data from our example (in table form) might look like the following:

Table 3

Feature Engineering Example

student_ID	number_rec_quest_score	avg_counting_quest_score
A1	340	131
A2	635	607
A3	601	720
…	…	…

In general, there are a range of mathematical operators to use when engineering features, and applying them systematically when paired with human judgement can produce strong feature sets for analysis in educational data mining (e.g. Paquette et al., 2014; Malkiewich et al., 2016). These mathematical operations (e.g. standard deviation, average, median, mode, range, sum, difference/delta, etc.) can be applied to "base" features (element 2 & 4, Fig. 7) to create new ones. New features can also be created by applying a sequential pathway lens—for example, creating variables that represent sets of 3 or 4 actions in a row (sometimes referred to as n-grams); creating new user groups based on progress pathways (e.g. "game finishers", "game quitters—early game", "game quitters—mid game"; Owen, 2014); and simply applying the lens "last," which might record the last interaction/objective/score (etc.) a player had before an important event (e.g. before quitting the game).

After features are selected, aggregated, and engineered, the next step is to perform the analysis, evaluating with a metric when applicable, in this case correlation (element 6, Fig. 7). The findings (element 7) are the results of the analysis interpreted for insight into the original research question. These findings can feed back into an iterative process by giving insight into new feature engineering, or different algorithm selection (elements 3-7, Fig. 7). In our example, we may get a small correlation which is non-significant after post-hoc correction for computing several correlations (although for sufficiently large data sets, all statistical tests will come out significant; magnitude of correlation then becomes a better metric), and find that score average is not a good measure after all, which might lead to consideration of different features for engineering. Or, it may spur the selection of a different algorithm (e.g. association rule mining, or a complex regression algorithm such as random forest). In many cases, initial findings may even feed into new research questions (e.g. "If there is no association between math quests with related math skills, are our game quests actually measuring math skills at all?").

3.3 Summary: Feature Engineering & Analysis

Overall, as this process supports iterative feature selection, engineering, and analysis, it can optimize insight into player behavior through increasingly nuanced, informed features and model development. This serves to uncover emergent patterns in the data that can prove critical to learning trajectories, particularly in complex systems like games and simulations. The deeper these insights are into player behavior in serious games and computer-based playful learning, the more robust and nuanced player profiles built on player interaction can be.

Our example above is a simple one, but as games become more complex, and research questions become more in-depth, analyses can involve more sophisticated algorithms (e.g. cluster analysis, predictive modeling with tree methods, and detector models which utilize multiple data sources). As this happens, feature selection, aggregation, and engineering tend to become more intricate and complex as well. The following extended example of *Progenitor X* explores base feature aggregation and feature engineering to build a predictive detector model of student behavior within the game.

3.4 *Progenitor X*: Feature Engineering for Optimal Insight

With Progenitor's game design, data framework design and base data stream established earlier (green design section, Fig. 1), the following paragraphs explore corresponding base feature aggregation and event-stream feature engineering (elements 4-5, Fig. 7). This section references recent research (Owen et al., 2016), using the established study as an example application of the current paper's feature engineering process. As a backdrop to this, we'll discuss the study's research question and aligned algorithm (element 3). Because the original study was focused on mining behavior inherent to the medium of games (which invites playful boundary testing as a means for discovering underlying rule systems; Salen & Zimmerman, 2004), it asked the research question: "What interactions with the game characterize thoughtful exploration and boundary testing?" To answer this question, a behavior detector of thoughtful exploration in *Progenitor X* was built using M5', a decision tree algorithm used for numerical prediction (cf. Breiman et al., 1984; Wang & Witten, 1997) which required intricate event-stream features as input variables for analysis. The corresponding event-stream data and feature creation will be highlighted here as an in-depth example of feature engineering in our main ecology (elements 4-5, Fig. 7).

3.4.1 Zombie Data: Base Feature Selection & Aggregation

To recap briefly, *Progenitor X* is a zombie-themed biology game, and provides a virtual lab environment in which students cultivate healthy cells, tissue and organs to cure the epidemic. As such, the experience is organized into nested progress markers (or "units", see Fig. 6): the game, objectives, and cycles (Fig. 5).

Thus, feature selection for input into the M5' model was organized around these units (cycles and larger objectives) as markers of core progress through the game. Within these game units, the features selected included virtually all Player Action and System Events captured by ADAGE (Fig. 6). A broad range of base features was desirable, since almost any interaction could be important to the target behavior; in addition, models of this type are well-suited to large volumes of event-stream data, having the capacity to intake a large number of variables and prune them to a well-defined pool of predictors. Table 4 visualizes this matrix of final features, showing the game units (objectives and cycles)

corresponding to event-stream interactions, organized by progression (all movement within the game) and performance (success and failure in the game). Features include objective/cycle starts and ends, objective/cycle success and failure, UI buttons used (some are optional during an objective, and others required), play duration, kind of cell used during a cycle (including a beginner "cell type I", and a more advanced "cell type II"), and the amount of health the lab cells lose in one cycle (the more a player treats the cells, the more damage they take). Note that these features are different in kind (and finer-grained) than the features used in Table 1; with complex learning games, the same data stream can often be sliced and represented in several different fashions. This is one of the key reasons why feature selection and engineering is needed; the data cannot simply be thrown into an algorithm, it must be transformed first so that the right aspects of an initially highly complex data stream are considered.

Table 4

Base data feature selection

	Progression	*Performance*
Objective	• objective added • objective starts • objective ends • UI tab: mission screen use • mission screen: city selected • UI button use: view objective • UI button use: add objective • objective type: cell, tissue, or organ	• objective successes • UI event: objective complete • number of cycles in a completed objective • objective failures • objective restarts • objective attempt number
Cycle	• cycle starts • cycle ends • cycle quits • cycle type: cell, tissue, organ • type of cell started in cycle (type I or II) • type of cell collected in cycle (type I or II) • UI tool select: "start" • UI tool select: "move" • UI tool select: "treatment" • UI tool select: "collect" • UI tool use: "start" • UI tool use: "move" • UI tool use: "treatment" • UI tool use: "collect"	• cycle successes • number + type of cells collected • number of turns in a cycle • health remaining at end of cycle • failure (health runs out) • failure (wrong cell collect) • % of health used in each cycle
Overall	• time elapsed (seconds) • game starts • UI button use: view next instruction ("next") • UI button use: view last instruction ("back") • UI button use: sound off/on • UI button use: almanac • Lab grid select: cell, tissue, or organ	• game win/loss • mission success events • total number of objectives complete • total number of missions complete • mission quit • game quit • UI button use: mission/game restart

These features give high-resolution information on student interaction throughout every phase of play, as well as a view into performance on designed learning mechanics. These base features became a foundation for the analyses to follow. Specifically, the aggregation of these features (element 5, Fig. 7) at the student level became the base input variables into the predictive model of thoughtful exploration.

3.4.2 Feature Engineering in Progenitor X

Before we attempted to build a model on the data, these variables underwent additional feature engineering to enhance capture of high-resolution player interaction. Specifically, the data transformations (which include mathematical operators as well as sequence lenses) expressed in Table 5 were performed on the existing base features to create new variables. The resultant new variables were created at the student level (adding fields to a data frame with one row per student).

Table 5

Data Transformations for Feature Engineering: Progenitor Examples

I) Data Transformation Examples	**II) Sample Features**
• **Totals/counts by event** (overall, for each objective, and for each cycle type)	- total failures in objective 1 - total tissue cycle failures - total objective 3 successes - total failures (whole game)
• **Ratios** (proportions of performance features)	- tissue failures: cell failures - total game successes: total game failures - tutorial objective failures: total failures
• **Averages** (per objective & per cycle type)	- average failure per objective completed - average success per objective added
• Performance data for **last objective played**	- successes in last played objective - failures in last played objective - name of last objective played
• **Chronological sequence** (taken as a sequence of data points, by objective & by cycle type)	- list of # of failures: objectives 1, 2, and 3 - list of # of successes: objectives 6, 7, & 8 - list of # of failures in tutorial levels only: objectives 1, 2, and 4
• **Isomorphic sequence** (for identical cycles only)	- identical cell cycle successes (from objectives 2, 5, and 8) - identical tissue cycle failures (from objectives 3, 4, and 7)

Finally, the event-stream features for input into the model captured over eighty game interactions at the student level. Iterations of feature engineering occurred in preparation (cycles of elements 5, 6, and 7, Fig. 7). During the early rounds of analyses, certain input variables like success in the late game (the final, most difficult boss level) emerged as predictive of the target behavior (thoughtful exploration), and subsequent features were engineered to provide higher resolution on boss level performance (e.g. percent of successes in the boss level, and number of successes in the most difficult half of the boss level). When final analysis was performed (element 7), these features were put into the final M5' model with the

established study's behavior label as an outcome variable (Owen et al., 2016). The model results are discussed in the next section, as well as contrasted with results from a comparable analysis which utilized far less rigor in feature engineering processes.

4 Empirical Results: Optimizing Behavioral Insights through IDEFA

This section gives an empirical demonstration of the design-integrated, analysis-informed feature engineering process presented in this paper. As an extension of the *Progenitor X* example based in a recent study (Owen et al., 2016), a model is reviewed which has utilized the feature engineering process to optimize analysis input data. This final model (we will call it the Feature Engineered or *FE Model*) is contrasted with a similar analysis model which did not use the same integrated feature engineering process, but relied instead on lower resolution data (we will call the latter the *RD Model*, where RD stands for Raw Data).

4.1 Results: FE Model

The FE Model shows the final behavior detector generated from the *Progenitor* example detailed throughout this paper. As a result of the interplay between base feature selection, aggregation, feature engineering and iterative analysis, the final *Progenitor* behavior detector achieved strong predictive performance. As summarized from the original study (Owen et al., 2016), the final behavior detector used M5', a decision tree model for numerical prediction. M5' trees are a common data mining method for making numerical prediction when the relationships being modeled are partially non-linear; they capture both linear relationships and non-linear relationships, unlike other approaches which can only capture one of these two types of relationships. Generally, the ability to partition data and effectively prune a large number of potential predictors make M5' trees suitable for dealing with large, complex data sets as well as relatively smaller data sets. M5' trees produce a predicted value for a numerical variable, in this case the number of thoughtful exploration instances, as follows:

- First, a set of "if-then-else" decisions are made. In each of these "if-then-else" decisions, the tree asks if a specific variable's value is greater than or less than a benchmark value. This is visually shown as a left and right fork (path) in the tree structure – see Figure 8 for an example. If the variable's value is less than or equal to the benchmark, the left fork is chosen. If it is greater, the right fork is chosen. A different set of future decisions follows each decision based on whether the left or right fork is chosen. These decisions capture the non-linear relationships in the data.
- After a set of "if-then-else" decisions, a linear regression equation is used on other variables. Different equations are used at the end of different paths through the tree. These equations capture the linear relationships in the data. The linear regression equation produces a final value for the predicted variable, in this case the number of thoughtful exploration instances.

The engineered event-stream features produced in our process served as input variables, and target behavior instances as an outcome variable. (These behavior instances of thoughtful exploration (TE) were coded using text replays (e.g. Baker & de Carvalho, 2008), and since weight was given to how *much* players engaged in thoughtful exploration as part of a full play trajectory, the predicted variable was the number of instances of TE for each student.) The model achieved a cross-validated correlation of .627 (r^2 = .393, indicating that this model as a whole explains 39.3% of the variance in the occurrence of

thoughtful exploration), comparable to levels in similar game-based learning detector models (e.g. Baker & Clarke-Midura, 2013). Predictors were iteratively developed features from Table 4 and Table 5, using the interconnected, iterative design-data-analysis approach presented in this paper. This level of resolution into user interactions enhanced insight into player trajectories, with the M5' output identifying three branches of students, grouped into three predictive models of behavior. There is some degree of collinearity in these predictors. Whereas many statisticians advocate the elimination of all significant collinearity, data mining algorithms often exploit collinearity to find complex relations between variables to predict variables of interest. While collinearity can reduce interpretability, entirely eliminating correlated predictors variables can in some cases reduce the potential to find complex patterns that better explain the data. Ultimately, the test of a data mined model is its ability to predict new data (tested here through cross-validation); if collinearity leads to model over-fitting (fitting to noise rather than signal), performance on new data will suffer. As such, a moderate approach is often taken towards collinearity by data miners – treating collinearity as a potential empirical concern and limiting factor rather than as a situation to be avoided regardless of the impact on predictive accuracy.

The final result of three branching prediction models, with detailed features, follows below:

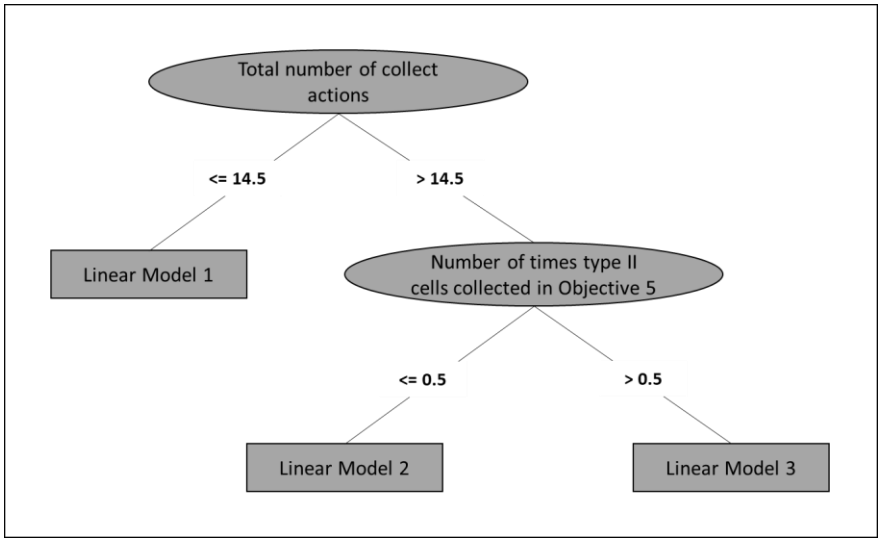

Fig. 8 A branching tree view of the FE model results.

Total number of collect actions <= 14.5 : Linear Model 1 (63/55.337%)
Total number of collect actions > 14.5 :
| Number of times type II cells collected in Objective 5 <= 0.5 : Linear Model 2 (32/51.697%)
| Number of times type II cells collected in Objective 5 > 0.5 : Linear Model 3 (15/48.256%)

Linear Model 1:
number of total Thoughtful Exploration instances =
*0.0011 * duration (seconds) of Obj 0 (training)*
*+ 0.0282 * total number of cells collected in Obj 0*
*+ 0.7095 * average % health used in Obj 0*
*+ 0.0256 * total times optional UI buttons used in Obj 0*
*+ 0.0026 * duration (seconds) of Obj 1*
*- 0.0178 * number of type I cell cycles in Obj 1*

*- 0.2307 * average % health used in Obj 2 (first half)*
*+ 0.0007 * duration (seconds) of Obj 2 (last half)*
*- 0.196 * average % health used in Obj 2 (last half)*
*+ 0.104 * number of times type II cells collected in Obj 5*
*+ 0.0022 * total number of cells collected in Obj 5*
*+ 0.0957 * number of type II cell cycles in Obj 8*
*- 0.0033 * duration (seconds) of Obj 8 (last half)*
*+ 0.1094 * number of successful cycles in Obj 8 (last half)*
*- 0.0076 * total number of cell or tissue collection instances*
- 0.1585

Linear Model 2:
number of total Thoughtful Exploration instances =
*0.8926 * average % health used in Obj 0*
*+ 0.0322 * total times optional UI buttons used in Obj 0*
*+ 0.0008 * duration (seconds) of Obj 1*
*- 0.1119 * number of type I cell cycles in Obj 1*
*+ 0.0033 * duration (seconds) of Obj 2 (last half)*
*- 0.2903 * average % health used in Obj 2 (first half)*
*+ 0.0009 * duration (seconds) of Obj 2 (last half)*
*- 1.4842 * average % health used in Obj 2 (last half)*
*+ 0.321 * number of times type II cells collected in Obj 5*
*- 0.0143 * total number of cell collection instances in Obj 5*
*+ 0.0027 * total number of cells collected in Obj 5*
*+ 0.2328 * number of type II cell cycles in Obj 8*
*- 0.0041 * duration (seconds) of Obj 8 (last half)*
*+ 0.1377 * number of successful cycles in Obj 8 (last half)*
*- 0.0095 * total number of cell or tissue collection instances*
+ 2.3512

Linear Model 3:
number of total Thoughtful Exploration instances =
*- 0.196 * number of type I cell cycles in Obj 1*
*+ 0.8926 * average % health used in Obj 0*
*+ 0.0322 * total times optional UI buttons used in Obj 0*
*+ 0.0008 * duration (seconds) of Obj 1*
*- 0.0687 * number of type I cell cycles in Obj 1*
*- 0.0547 * number of type II cell cycles in Obj 2*
*- 0.2903 * average % health used in Obj 2 (first half)*
*+ 0.0009 * duration (seconds) of Obj 2 (last half)*
*- 0.7426 * average % health used in Obj 2 (last half)*
*+ 0.4288 * number of times type II cells collected in Obj 5*
*- 0.0224 * total number of cell collection instances in Obj 5*
*+ 0.0027 * total number of cells collected in Obj 5*
*+ 0.5849 * number of type II cell cycles in Obj 8*
*- 0.0041 * duration (seconds) of Obj 8 (last half)*
*+ 0.1377 * number of successful cycles in Obj 8 (last half)*
*- 0.0095 * total number of cell or tissue collection instances*
+ 2.7171

Initially, the results are split into three predictive models by the total number of collect actions (the final step a player performs in the start/treat/collect actions to finish a game cycle—see Fig. 4). Consequently, this model shows interesting splits along the number of collect actions—suggesting finished/non-finished player groupings—and with models 2 and 3, splitting around the collection of advanced (type II) cells. The detector results therefore show three potential groupings of students in relationship to the behavior of thoughtful exploration, each with nuanced trajectories and patterns of interaction. The game's design allowed players freedom of choice, and therefore produced variation in event-stream features like the number of cycles, amount of time, amount of health used, etc. per student; relationships between these features also varied accordingly—making the differentiation of three student groups even more meaningful. Students within each group have similar play trajectories relative to thoughtful exploration; each of these three groups could serve as a foundation for play profiling around boundary testing and productive failure (Kapur, 2006), for instance. Other behaviors of interest relative to user profiles could be studied with similar methods, and used to help establish a baseline of play styles and learner grouping derived from emergent player patterns. Thus, these kinds of EDM models—especially when supported with IDEFA processes of design-aligned feature engineering and iterative analysis—can enhance insight into behavior and foundational player profiles.

4.2 RD Model: Results and Comparison

In contrast to this relatively sophisticated model, a parallel model of behavior was built which did not utilize the rigorous feature engineering process outlined in this study. This model retained the same behavior label outcome variable, and used the same M5' algorithm; however, the event-stream input variables were limited to basic completion and time information, shown in Table 6. This information showed basic progression through the game, but did not offer a detailed view into performance (i.e. was devoid of successes and failures, which are critical components of "results" in the core data framework, element 1), progress marker specifics (e.g. whether a cycle was a cell, tissue or organ cycle, or what cell types were involved), or detailed player action (also a central to a strong data framework, such as UI interaction, or core laboratory tool use). Essentially, these features underwent minimal base selection (element 4), no feature engineering (element 5), and were not improved through iterative development (cyclical elements 4-7).

Table 6

Basic input variables for analysis Model RD

Objective	*Time*	*Progress*
Tutorial Objective 0	Seconds spent in Obj 0	Number of cycles completed in Obj 0
Objective 1	Seconds spent in Obj 1	Number of cycles completed in Obj 1
Objective 2	Seconds spent in Obj 2	Number of cycles completed in Obj 2
Objective 3	Seconds spent in Obj 3	Number of cycles completed in Obj 3
Objective 4	Seconds spent in Obj 4	Number of cycles completed in Obj 4
Objective 5	Seconds spent in Obj 5	Number of cycles completed in Obj 5
Objective 6	Seconds spent in Obj 6	Number of cycles completed in Obj 6

Objective 7	Seconds spent in Obj 7	Number of cycles completed in Obj 7
Objective 8	Seconds spent in Obj 8	Number of cycles completed in Obj 8
Total Game	Seconds spent in total game	Number of cycles completed in game
		Number of obj completed in game

This contrasting model was run with the same M5' algorithm and identical outcome variable as the previous analysis. It was also cross-validated at the student level and measured with the same goodness metric, linear correlation. However, it only achieved a much lower correlation of .28 (r^2 = .078, indicating that this model as a whole explains 7.8% of the variance in the occurrence of thoughtful exploration), less than half the value of the feature engineered model. The model results are as follows:

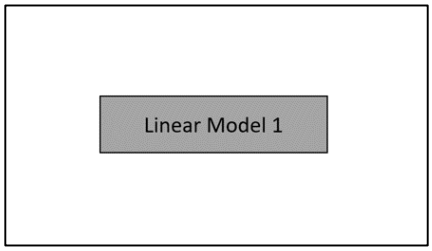

Fig. 9 A zoomed-out view of the single-branched RD tree results.

In all cases use Linear Model: 1
Linear Model: 1

*0.0029 * Seconds spent in Obj 2*
*- 0.0022 * Seconds spent in Obj 5*
*- 0.2021 * Number of cycles completed in Obj 7*
*- 0.0502 * Number of cycles completed in Obj 2*
*+ 0.097 * Number of cycles completed in Obj 5*
*+ 0.0019 * Total_time*
+ 0.1689

Interestingly, only one model predicting the target behavior was produced in this version, producing less nuanced insight into branches of player behavior. It should be noted that a simple M5' model with a single branch is not problematic by definition; however, the low cross-validated correlation, and low interpretability and meaningfulness of the resultant model, speak to this model's inferior resolution of event-stream input data.

Overall, the better results of the three-pronged M5' model based on feature engineering show the considerable impact of a rigorous feature engineering process—rooted in strong data framework design, fueled by clear feature selection and methodical lenses for feature engineering, and ultimately honed by iterative investigation– which empirically produce insight into player behavior.

5 Discussion and Conclusion

This article puts forward a systemic approach to robust feature engineering for optimal behavior insights in serious games. The interconnectedness of game design, data frameworks, and data capture (Part I) provide a foundation for the subsequent feature selection, aggregation and feature engineering integrated

with iterative analysis (Part II). Empirical investigation into application of this IDEFA process demonstrates the ability of this integrated, iterative approach to feature engineering to optimize insights into player behavior. Results show that a model developed based on raw data not enhanced through feature engineering applied performed poorly in comparison to a model incorporating full iterative feature engineering. The classification model used with engineered features showed more meaningful results, with greater insight into specific game features that impacted player decisions, as well highlighting several trajectories of play that characterized groups of players. These results surface emergent patterns of player interaction, providing a strong base for creating behavior-based play profiles that are more data-driven (derived from organic patterns in play).

IDEFA's integrated process for optimized feature engineering outlines a systematic approach to creating high-quality event-stream variables for use in learning analytics and educational data mining. To illustrate the application of this method to predictive modeling of game behavior (a base for building player profiles), the paper uses an example of behavior detection from a recent study (Owen et al., 2016), but this generalized process is designed to support many kinds of analysis. In particular, any analysis which uses event-stream data for insight into user interactions can benefit from this process—including visualization, structure discovery (e.g. cluster analysis), prediction, relationship mining (e.g. correlation-based analyses and rule mining), and behavior detection. Future studies in implementation of this work in these broader analysis categories would support further refinement of the IDEFA framework, and illustrate broader application of an integrated feature engineering process in service of learner insights.

Applying the IDEFA process to multiple analyses and game contexts also empowers data-driven design. Precise, design aligned data allows for clear interpretation of event-stream features, and implementation of a structured data framework (element 1) supports consistent labeling of data across games. This is particularly useful in large-scale systems which use multiple games to teach interwoven concepts or skills (e.g. Age of Learning[6], or GlassLab Games[7]). As analysis is completed with these clear, consistent features, there is potential for insights in user behavior across the system—and an opportunity to iteratively improve design based on emergent player patterns.

Overall, integrating feature engineering with design, data collection, and iterative analysis can result in variables which increase insight into user behavior. The IDEFA process gives a systematic approach to fueling these insights into player decisions, key to behavior-based profiles of play. Fueled by a design-aligned base data-stream, application of multiple computational lenses, and iteration with analysis, this rigorous feature engineering process provides a strong foundation for gaining insight into player behavior in immersive digital learning environments.

References

Baker, R. (2010). Data mining for education. *International Encyclopedia of Education, 7*(3), 112–118.

Baker, R. (2013). *Big Data in Education--Week Six: Feature Engineering*. Retrieved from www.educationaldatamining.org/bde/W003V001v1.pptx

Baker, R., Gowda, S., & Corbett, A. (2011). Towards predicting future transfer of learning. In *Artificial intelligence in education* (pp. 23–30). Springer. Retrieved from http://link.springer.com/content/pdf/10.1007/978-3-642-21869-9.pdf#page=49

Baker, R., & Clarke-Midura, J. (2013). Predicting Successful Inquiry Learning in a Virtual Performance Assessment for Science. In *Proceedings of the 21st International Conference on User Modeling,*

[6] http://www.ageoflearning.com/

[7] https://www.glasslabgames.org/

Adaptation, and Personalization (pp. 203–214). Retrieved from http://www.columbia.edu/~rsb2162/UMAP-2013-BCM-v9.pdf

Baker, R., & de Carvalho, A. (2008). Labeling student behavior faster and more precisely with text replays. In *Proceedings of the 1st International Conference on Educational Data Mining* (pp. 38–47). Retrieved from http://learnlab.org/uploads/mypslc/publications/edm2008textreplayalgebrag.pdf

Baker, R., & Siemens, G. (2014). Educational data mining and learning analytics. In K. Sawyer (Ed.), *Cambridge Handbook of the Learning Sciences* (2nd Edition, pp. 253–274).

Berland, M., Baker, R., & Blikstein, P. (2014). Educational data mining and learning analytics: Applications to constructionist research. *Technology, Knowledge and Learning*, *19*(1–2), 205–220.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks.

Chung, G. K. W. K. (2015). Guidelines for the design, implementation, and analysis of game telemetry. In C. S. Loh, Y. Sheng, & D. Ifenthaler (Eds.), *Serious games analytics: Methodologies for performance measurement, assessment, and improvement* (pp. 59–79). New York, NY: Springer.

Clark, D. B., Martinez-Garza, M. M., Biswas, G., Luecht, R. M., & Sengupta, P. (2012). Driving Assessment of Students' Explanations in Game Dialog Using Computer-Adaptive Testing and Hidden Markov Modeling. In *Assessment in Game-Based Learning* (pp. 173–199). New York: Springer.

Corrigan, S., DiCerbo, K. E., Frenz, M., Hoffman, E., John, M., & Owen, V. E. (2015, February 5). *GlassLab Game Design Handbook*. GlassLab Games, Redwood City, CA. Retrieved from http://gamedesign.glasslabgames.org/

Danielak, B. (2014). *Analyzing Data with ADAGE*. Gitbooks. Retrieved from https://capbri.gitbooks.io/makescape-adage-gitbook/

DiCerbo, K. E., & Kidwai, K. (2013). Detecting Player Goals from Game Log Files. Presented at the 6th International Conference on Educational Data Mining. Retrieved from http://www.educationaldatamining.org/EDM2013/papers/rn_paper_58.pdf

Fogarty, J. A. (2006). *Constructing and evaluating sensor-based statistical models of human interruptibility*. IBM Research. Retrieved from http://www.cs.cmu.edu/afs/.cs.cmu.edu/Web/People/jfogarty/publications/jfogarty-dissertation-final.pdf

Fogarty, J., Hudson, S. E., & Lai, J. (2004). Examining the robustness of sensor-based statistical models of human interruptibility. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 207–214). ACM.

Frank, E., Hall, M., Trigg, L., Holmes, G., & Witten, I. H. (2004). Data mining in bioinformatics using Weka. *Bioinformatics*, *20*(15), 2479–2481.

Games Learning Society. (2012). *ADAGE*. Madison, WI: University of Wisconsin-Madison. Retrieved from http://www.gameslearningsociety.org/

Gee, J. P. (2003). *What Video Games Have to Teach Us About Learning And Literacy*. Palgrave Macmillan.

Gee, J. P. (2005). Learning by design: Good video games as learning machines. *E-Learning and Digital Media*, *2*(1), 5–16.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, *3*, 1157–1182.

Habgood, M. J., & Ainsworth, S. E. (2011). Motivating children to learn effectively: Exploring the value of intrinsic integration in educational games. *The Journal of the Learning Sciences*, *20*(2), 169–206.

Halverson, R., & Owen, V. E. (2014). Game Based Assessment: An Integrated Model for Capturing Evidence of Learning in Play. *International Journal of Learning Technology [Special Issue: Game-Based Learning]*, *9*(2), 111–138.

Hao, J., Smith, L., Mislevy, R., von Davier, A., & Bauer, M. (2016). *Taming Log Files From Game/Simulation-Based Assessments: Data Models and Data Analysis Tools* (ETS Research Report Series) (pp. 1–17).

Ifenthaler, D., Adcock, A. B., Erlandson, B. E., Gosper, M., Greiff, S., & Pirnay-Dummer, P. (2014). Challenges for Education in a Connected World: Digital Learning, Data Rich Environments, and Computer-Based Assessment—Introduction to the Inaugural Special Issue of Technology, Knowledge and Learning. *Technology, Knowledge and Learning*, *19*(1–2), 121–126. https://doi.org/10.1007/s10758-014-9228-2

Kapur, M. (2006). Productive failure. In S. Barab, K. Hay, & D. Hickey (Eds.), *Proceedings of the International Conference on the Learning Sciences* (Vol. 0, pp. 307–313).

Kevan, J. M., & Ryan, P. R. (2016). Experience API: Flexible, Decentralized and Activity-Centric Data Collection. *Technology, Knowledge and Learning*, *21*(1), 143–149. https://doi.org/10.1007/s10758-015-9260-x

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The Knowledge-Learning-Instruction Framework: Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cognitive Science*, *36*(5), 757–798. https://doi.org/10.1111/j.1551-6709.2012.01245.x

Loh, C. S. (2012). Information Trails: In-process assessment of game-based learning. In *Assessment in Game-Based Learning* (pp. 123–144). New York: Springer.

Malkiewich, L., Baker, R., Shute, V. J., Kai, S., & Paquette, L. (2016). Classifying behavior to elucidate elegant problem solving in an educational game. In *Proceedings of the 9th International Conference on Educational Data Mining* (p. 448). Raleigh, NC.

Mislevy, R. J., & Haertel, G. D. (2006). Implications of Evidence Centered Design for Educational Testing. *Educational Measurement: Issues and Practice*, *25(4)*, 6–20.

Newell, A. (1973). Visual Information Processing. In W. G. Chase (Ed.). New York: Academic Press.

Owen, V. E. (2014). *Capturing In-Game Learner Trajectories with ADAGE (Assessment Data Aggregator for Game Environments): A Cross-Method Analysis*. University of Wisconsin-Madison, Madison, WI.

Owen, V. E., Anton, G., & Baker, R. (2016). Modeling User Exploration and Boundary Testing in Digital Learning Games. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization* (pp. 301–302). New York, NY: ACM Press.

Owen, V. E., Shapiro, R. B., & Halverson, R. (2013). Gameplay as Assessment: Analyzing Event-Stream Player Data and Learning Using GBA (a Game-Based Assessment Model). In *CSCL 2013 Conference Proceedings* (Vol. Volume 1 — Full Papers & Symposia, pp. 360–367). Madison, Wi: International Society of the Learning Sciences (ISLS).

Paquette, L., de Carvahlo, A., Baker, R., & Ocumpaugh, J. (2014). Reengineering the Feature Distillation Process: A case study in detection of Gaming the System. In *Educational Data Mining 2014*.

Retrieved from http://www.educationaldatamining.org/conferences/index.php/EDM/2014/paper/download/1447/1413

Salen, K., & Zimmerman, E. (2004). *Rules of play: Game design fundamentals.* MIT Press.

Sao Pedro, M. A., Baker, R., & Gobert, J. D. (2012). Improving construct validity yields better models of systematic inquiry, even with less information. In *International Conference on User Modeling, Adaptation, and Personalization* (pp. 249–260). Springer. Retrieved from http://link.springer.com/10.1007%2F978-3-642-31454-4_21

Serrano-Laguna, A., Martinez-Ortiz, I., Haag, J., Regan, D., Johnson, A., & Fernández-Manjóna, B. (2017). Applying standards to systematize learning analytics in serious games. *Computer Standards & Interfaces*, *50*, 116–123.

Shute, V. J. (2011). Stealth assessment in computer-based games to support learning. *Computer Games and Instruction*, *55*(2), 503–524.

Shute, V. J., & Kim, Y. J. (2014). Formative and Stealth Assessment. In J. M. Spector, M. D. Merrill, J. Elen, & M. J. Bishop (Eds.), *Handbook of Research on Educational Communications and Technology* (pp. 311–321). New York, NY: Springer New York. https://doi.org/10.1007/978-1-4614-3185-5_25

Squire, K. (2006). From content to context: Videogames as designed experience. *Educational Researcher*, *35*(8), 19–29.

Squire, K. (2011). *Video Games and Learning: Teaching and Participatory Culture in the Digital Age.* Teachers College Press. Retrieved from http://eric.ed.gov/?id=ED523599

Steinkuehler, C., Barab, S., & Squire, K. (Eds.). (2012). *Games, Learning, and Society: Learning and Meaning in the Digital Age*. New York: Cambridge University Press.

Wang, Y.-C., & Witten, I. H. (1997). Inducing model trees for continuous classes. In *Proceedings of the Ninth European Conference on Machine Learning* (pp. 128–137). Retrieved from http://www.cs.waikato.ac.nz/~ml/publications/1997/Wang-Witten-Induct.pdf