Game Analytics: User Engagement and Performance

Center for Data Science, New York University

DSGA-1007: Programming for Data Science in Python

Professor Jacopo Cirrone

December 6, 2021

Group 15

Chinelo Ibekwe, Doma Ghale, Jing Zhang, Joseph Stansil

1. Introduction

Data comes from an experiment conducted in the fall of 2018 by the NYU CREATE Lab. In this experiment, 145 students played a game called All You Can E.T.. Their play generated 210k data points. The total amount of time of play was 2-3 hours.

The game aims to train cognitive flexibility. The game rules are explained in https://www.youtube.com/watch?v=EFiMlrgLRpo. For more details about the game and dataset, see the research paper by Zhang, Ober, Yang, Plass and Homer [1].

Variables in this dataset fall under the following categories (see Appendix A for details):

- Game context: gameCode, gameLevel, gameTime, alienID, alienTypeAndColor, speed, foodNeeded
- Player engagement: accessCode, userID, sesID, gsUserID, logTimestamp
- Player performance: hitPosX, hitPosY, reactionTime, hitType, foodShot

To evaluate player engagement and player performance, our inquiry is guided by the following questions:

1. How much have players played?
2. How to identify off-task players? (Off-task players are those who were observed to be not playing the game for a significant period during the experiment.)
3. How well did the players perform?
4. How many player clusters are there?

2. Data Analysis and Results

2.1 Data cleaning, wrangling and merging [2]

For data analysis we selected 20 files. Each file name comes with one of the five gameCode (each contains a group of variables) and one of the four accessCode (each contains a group of users). Facing the unique challenges (see Appendix C for details) with this dataset, we put a great effort and combined three gameCode (12 files) into one. Specifically, we did the following:

- Recode and rename columns: We converted the column gameLevel into two columns, which are level and attempt. Two gameCodes had two of the same column names, so we gave them more descriptive names.
- Fix the problem with logTimestamp: In this dataset, logTimestamp is the time when the server receives data. We noticed inconsistency in the logTimestamp for the same hit across different file types (i.e. gameCode), so we created a new logTimestamp using the timestamp of the first alien and the difference between the gameTime. We then used the new logTimestamp to calculate accuracy and find off-task players.
- Investigate and remove duplicate records: For example, the same hit was logged multiple times so we got rid of the rows if everything except for the logTimeStamp is the same.
- Missing or incomplete game levels: Using the expected number of aliens per level and number of distinct aliens logged for each gameID, we identified and removed records if the game level was not completed.

- Merge files: 3 out of 5 gameCode had manageable consistency in the gameTime, so we decided to not merge the other two files (see Appendix C for details).

2.2 Game analytics: player engagement

To answer the guiding questions, we conducted exploratory data analysis and asked ten specific questions. Below is our findings (see Appendix B for figures):

Question 1: How many players played each level? (level vs player) [3]

Generally as the level goes higher, there are fewer players who play it. The most popular level is level 11, as all 145 players played it. The highest level played is level 29.

There is a big jump between level 10 and level 11 because all players were mandatorily set to play level 11 in the middle of the experiment. The number of players who played level 2 is low because 80% of the level 2 attempts were incomplete, which we removed during the data cleaning process [2].

Question 2: A player can attempt a level multiple times. How many attempts were performed for each level? (level vs attempt) [3]

Generally, as the level goes higher, there are fewer attempts of this level. The most popular level measured by the number of attempts is level 3, which has 491 attempts.

Same explanation as Q2 for the big jump between level 10 and level 11 and a low number of attempts for level 2. It is interesting to see that the number of attempts for level 16 is unexpectedly high. This is worth further investigation.

Question 3: How many distinct levels did each user play? (player vs level) [3]

Most players played 5-20 distinct levels. On average, players played 12.9 levels.

Question 4: How many attempts per level of players? (player vs attempt/level) [3]

Most players played each level for 2-4 times. On average, players had 2.8 attempts per level.

Question 5: How do we classify the off-task players? [4]

In the context of this experiment, modelling the off-task players was a supervised learning task. During the study researchers individually noted down when they noticed a player was off-task and recorded their userID in a list, which we used to flag for ‘on-task’ in the data and perform several analyses on the correspondence of this flag.

We began by selecting indicators for the on or off-task behavior, including levels completed, reaction time, and accuracy at the player level. Additionally, we applied feature engineering to generate a column that reported the maximum gap in timestamps per user, to capture players looking away or otherwise not interacting with the game. We compared each indicator with the on/off task flag with Pearson correlational analysis. Intuitively, one would think that an off-task player completes fewer levels, or that clicks slowly, inaccurately, or less often. But there were no strong correlations found between the indicators and on-task flags.

Creating models with these indicators yielded similarly negative results. We used logistic regression on several subset of variables none of which performed substantially better than chance (~50- 60% accuracy on test data). We tried other modelling methods, including decision trees, stochastic gradient descent, and lasso regression, but they all had similarly middling results. It seems likely that, due to the inconsistent nature by which off-task users were flagged by the researchers, modelling it with the given data and matching their findings is intractable. One possible way to improve model performance is through rounds of deliberate feature engineering if we keep working on this question.

2.3 Game analytics: player performance

Question 6: What is the accuracy and reaction time of each level? (accuracy & reaction time, by level) [3]

Generally, the accuracy of each level is between 70% and 90%. The easiest level is level 29, as players hit an average of 92% of its aliens correctly. The most difficult level is level 16, as players hit an average of 73% of its aliens correctly.

Generally, the reaction time of each level is between 1s and 3s. The fastest level is level 8, as it took players 0.95s on average to hit each alien. The slowest level is level 27, as it took players 2.77s on average to hit each alien.

Question 7: What is the accuracy and reaction time of each player? What is the relationship between accuracy and reaction time? [3]

Generally, the accuracy of players is between 70% and 90%. The average accuracy of players is 82%. Generally, the reaction time of players is between 1s and 2.5s. The average reaction time of players is 1.58s. There is no linear relationship between accuracy and reaction time for players.

Question 8: What is the player accuracy along logtimeStamp? [5]

We plotted players' accuracy at different times of the day. Their accuracy in the afternoon is higher than that in the morning.

Question 9: What is the player reaction time along logtimeStamp? [5]

We plotted players' reaction time at different times of the day. They reacted faster in the afternoon than in the morning.

Question 10: How many player clusters are there? (K-means cluster analysis) [3]

We used K-means cluster analysis to classify players. We tried five meaningful combinations of four variables, which are player accuracy, player reaction time, max level played, and number of distinct levels played. We used silhouette analysis to determine the best number of clusters.

As a result, we can group players into two clusters based on their accuracy and reaction time. The silhouette coefficient is 0.61. Group 1 represents cautious players who have slightly higher accuracy and much higher reaction time than group 2.

3. Discussion, Conclusions and Future Work

Based on the results from exploratory data analysis, we draw the following conclusions.
For player engagement:

- Generally, as the level goes higher, there are fewer players who play it.
- Out of 50 levels, the highest level played is level 29.
- The most popular level is level 11. 80% of the level 2 attempts were incomplete.
- The most attempted level is level 3.
- Most players played 5-20 distinct levels. On average, players played 12.9 levels.
- Most players played each level for 2-4 times. On average, players had 2.8 attempts per level.

For player performance:

- The accuracy of each level is between 70% and 90%. The easiest level is level 29 (accuracy 92%). The most difficult level is level 16 (accuracy 73%).
- The reaction time of each level is between 1s and 3s. The fastest level is level 8 (reaction time 0.95s on average). The slowest level is level 27 (reaction time 2.77s on average).
- The accuracy of players is between 70% and 90%. The average accuracy of players is 82%.
- The reaction time of players is between 1s and 2.5s. The average reaction time of players is 1.58s.
- There is no linear relationship between players' accuracy and reaction time (see figure 7.3). This is supported by the theory of executive functions, which claims that accuracy and reaction time are two distinct performance dimensions.
- Players' performance is better in the afternoon than in the morning in terms of higher accuracy and lower reaction time .
- Players can be clustered into two groups based on their accuracy and reaction time. Group 0 is cautious players who have slightly higher accuracy and much higher reaction time while group 1 is balanced players (see figure 10).

The general trend of the reaction time matches our expectation that the higher levels are more difficult hence the players would take longer to react. However, players’ accuracy does not change drastically as the level goes higher, suggesting that players increase their reaction time to maintain their accuracy when they interact with more difficult levels.

In the future, we can answer questions such as are players’ accuracy and reaction time consistent over time? Can we predict the accuracy of a player based on how they performed in the past, combined with other variables such as gameLevel and type of alien? Can we use the log data to predict players’ post-test executive functions measured by a psychological task at the action level? What is the most difficult level given its accuracy and reaction time?

Appendix

A. Data dictionary

Variable Type	**Variable Name**	**Description**
game context	gameCode	Unique ID per a subset of variables
game context	gameLevel	50 possible game levels; format is x-y-z; level = 5x+y+1, attempt = z+1; example: SpaceCakesLevel 2-0-0
game context	gameTime	Created when the user launches the game and the time (in milliseconds) passed between it and when a new record is created
game context	alienID	The order alien spawned in a game level
game context	alienTypeAndColor	Eye number and face color of the alien
game context	speed	Speed at which alien falls (100 vs 125)
game context	foodNeeded	Type of food expected; sometimes FOOD/DRINK in complicated levels
player engagement	accessCode	Unique ID representing players in different schools or different grades
player engagement	userID	Unique ID assigned to each player
player engagement	sesID	Each session per week; a sesID is created every time the user logs onto the data collection platform
player engagement	gsUserID	A gsUserID is created each time a game is instantiated (after participants log onto the data collection platform). Each time a user opens a game a new gsUserID gets created.
player engagement	logTimestamp	Time when played as proxy; time when the server receives the data
player performance	hitPosX	X position where alien was hit; 0,0 is lower left corner
player performance	hitPosY	Y position where alien was hit; 0,0 is lower left corner
player performance	reactionTime	Time between when the first pixel of the alien appears on screen and when the player shoots
player performance	hitType	Three values: correct; wrong; missed. Use this variable to calculate accuracy; attention needed for FOOD/DRINK
player performance	foodShot	Type of food shot

B. Figures from game analytics

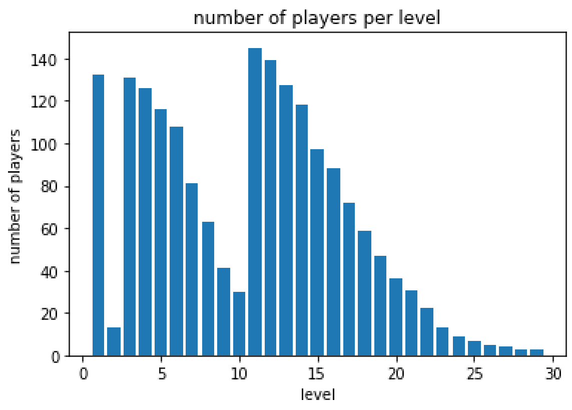


Figure 1.1

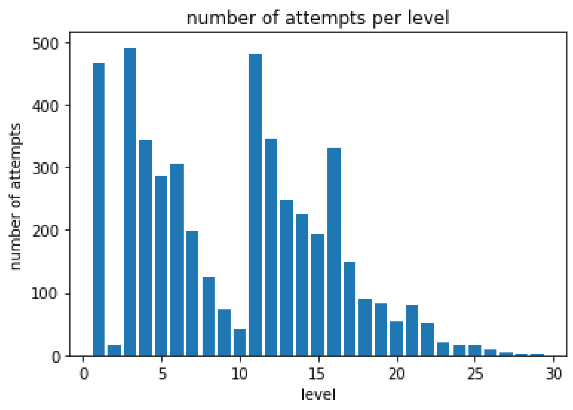


Figure 2

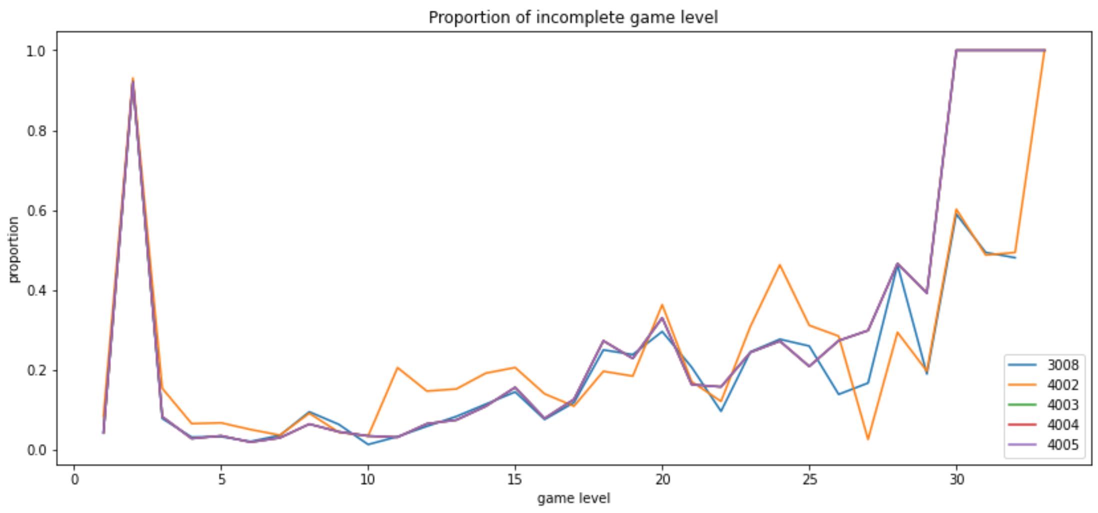


Figure 1.2 Proportion of incomplete game levels per game level

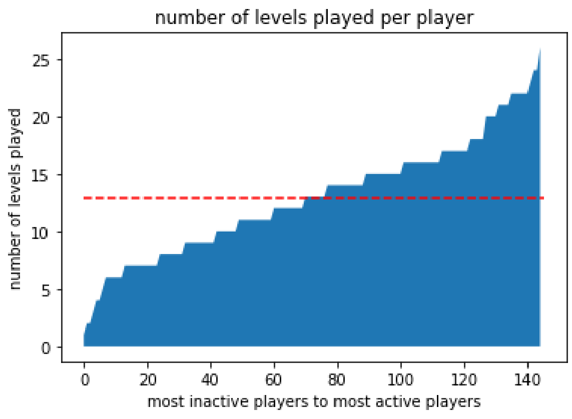


Figure 3

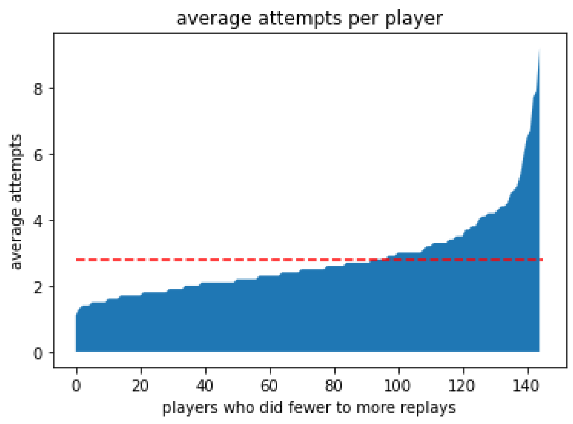
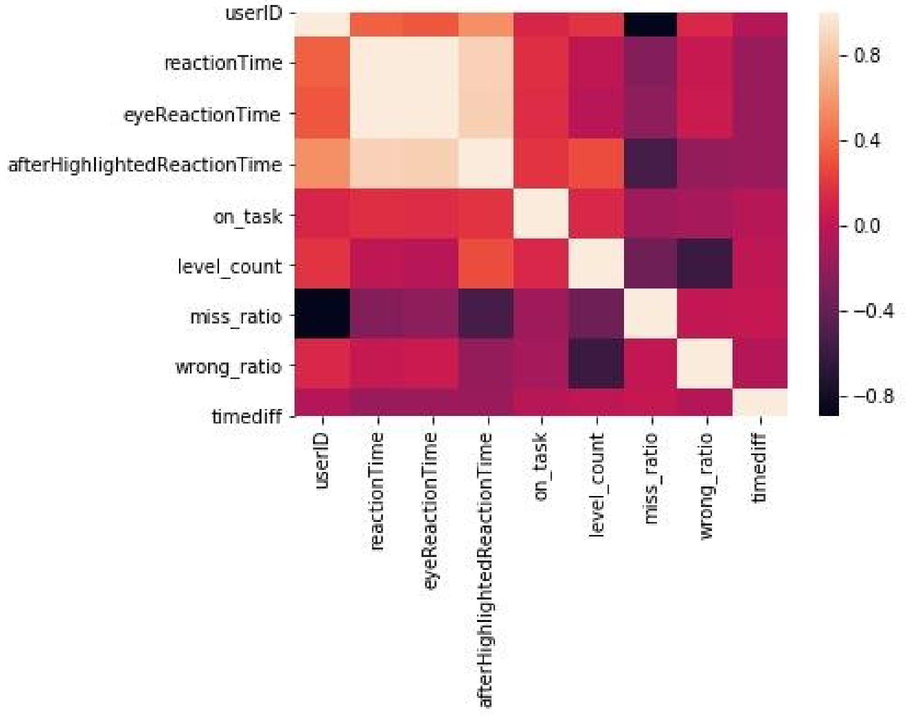


Figure 4



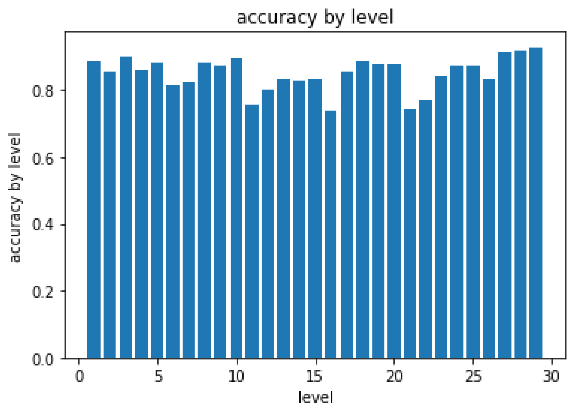
timediff - maximum gap in recorded timestamps

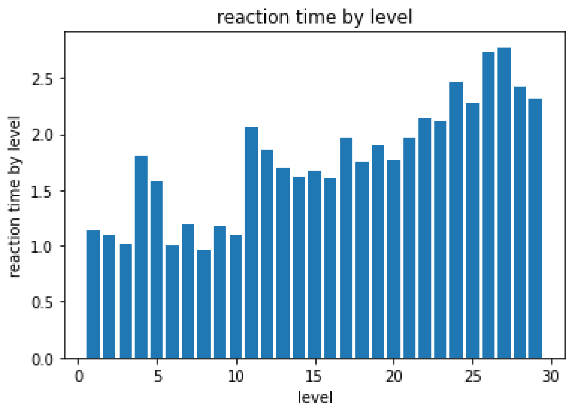
wrong_ratio and *miss ratio* – ratio of clicks that result in 'wrongs' or 'misses', i.e. accuracy

reactionTime, eyeReactionTime, and *afterHighlightedReactionTime* – recorded reaction times

level_count – number of levels completed

Figures 5





Figures 6.1

Figures 6.2

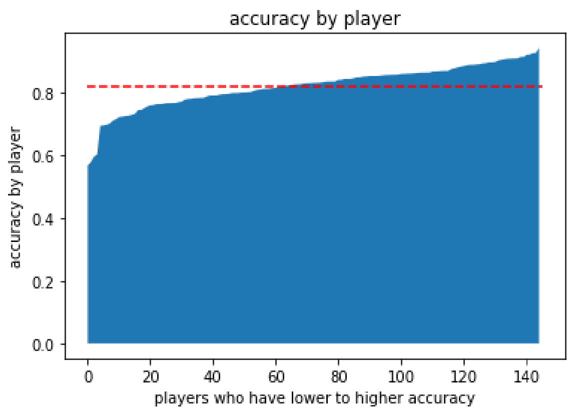


Figure 7.1

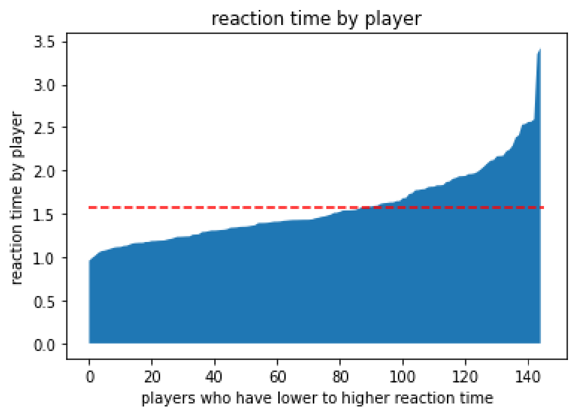
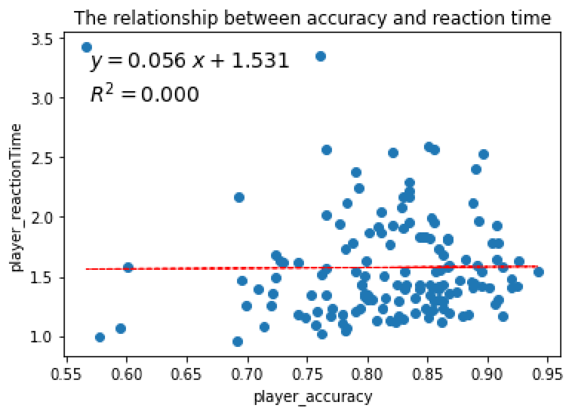


Figure 7.2



Figures 7.3

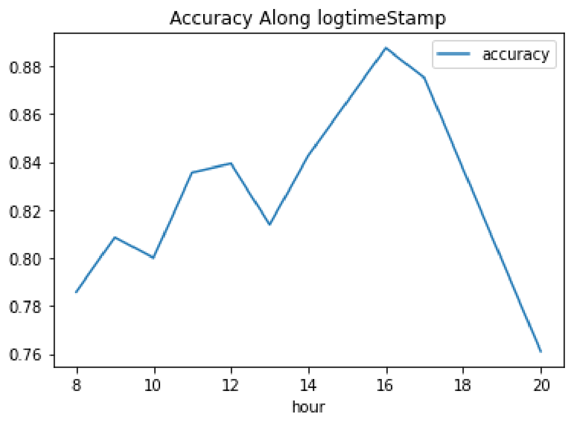


Figure 8

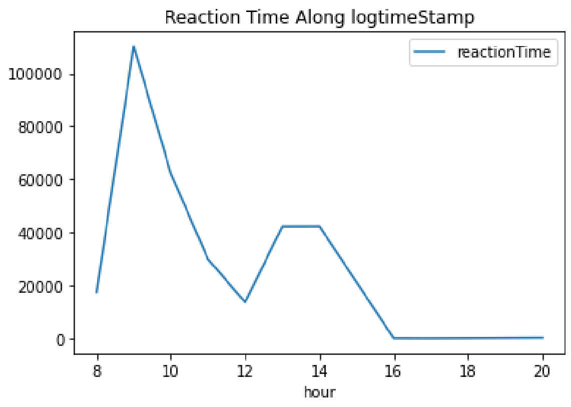


Figure 9

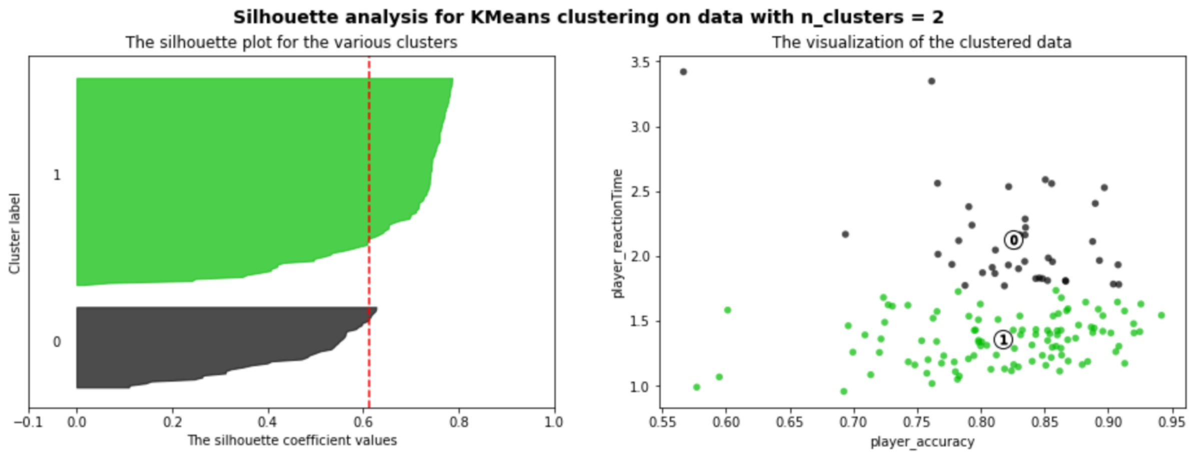


Figure 10

C. Challenges

C.1 Challenge with merging the files [2]

Initially, we selected 20 files (four accessCode per gameCode: 3008, 4002, 4003, 4004, 4005) that had the variables of our interest. Major challenge was merging files of different gameCode into one, which stemmed because of the lack of unique id that could have been used to relate the rows. Hence, we did a lot of data cleaning and manipulation to combine files. Below are some of the challenges we faced in this process:

1. There was inconsistency in log timestamp within a file and across files because it was the timestamp of when the database received the record not when the record was created. So, we decided to use the “gameTime”, which is how many milliseconds passed after the game was started till when each record was created. Not all gameTime records matched, but after small manipulations we were able to merge three (4003, 4004, 4005) of the five gameCode perfectly.
2. The number of rows in 3008 and 4002 were different from that of 4003, 4004 and 4005. 3008 has the spawn positions of aliens so it made sense that it would have more rows since an alien can be spawned and if the player doesn’t respond then the record for other gameCode wouldn’t be created. 4002 has information about the expected food and what the player shot. For some of the more complicated game levels players are expected to hit both buttons at once which creates two records instead of one, and was difficult to merge because of the first challenge mentioned above.
3. Some of the records had missing rows in one file but not others so we removed rows if the game level was incomplete (10% of the data). After removing the incomplete game levels, we expected the number of game levels played across different file types to match, but that was not the case. For example, “4002” contains the type of food that the alien needed and what was given by the user, and “4005” contains the hit position of the alien. So, we expected they would share the number of completed levels even though other file types may not.

C.2 Challenge with duplicate records [2, 6]

Initially, we thought for each (‘userID’, ‘sesID’), its (‘gameLevel’, ‘alienID’) would uniquely represent each player hit. We performed a duplicate check using those four variables and found 1/3 of the data were duplicate records. The problem is for each (‘userID’, ‘sesID’), there are duplicated gameLevel values for the same alienID. Again, gameLevel contains two pieces of information: the level and attempt. So, the problem seems to be that in each session, for each level, the attempt count is duplicated in 1/3 of the cases. We discussed the issue with the TA, within the group, and with the database manager from the lab. We decided to exhaust all other variables to see if there is any column that we missed that could explain the duplicate.

Contemplating the duplicate record, we noticed that a sesID had multiple gsUserID. For the same gsUserID, its [‘gameLevel’, ‘alienID’] would be unique. We learned from the database manager that sesID is created when the user launches the game platform while a gsUserID is created when the user further opens the game (a user can launch the same game multiple times after logging onto the platform). Thus, we decided to use gsUserID to replace sesID in the key, which greatly reduced the duplicate records (270 records). Though an explanation is needed as to why there is no unique ID to easily separate records for a gameLevel (includes the ith attempt) per user, we know these duplicate records were not generated by mistake, so we digress. We manipulated gameTime, so that (‘userID’, ‘gsUserID’, ‘gameTime’) could also be the key to merge the data.

References

1. Zhang, J., Ober, T., Yang, J., Plass J., & Homer, B. (2021). Predicting Executive Functions in a Learning Game: Accuracy and Reaction Time. In Proceedings of the 14th International Conference on Educational Data Mining (pp. 688-693).
2. Detailed Data Cleaning, Munging and Merging.ipynb
3. FinalProject_GameAnalytics_Jing.ipynb
4. Off Task Players.ipynb
5. Python group project code (chinelo, final).ipynb
6. FinalProject_DupCheck_Jing.ipynb