Capstone Data Wrangling
ACADEMIC MASTERY STUDY

All work described in this report can be viewed in the initial_data_exploration notebook.

Download **`grockit_all_data.zip`** data files from the Grockit competition on Kaggle, found here: https://www.kaggle.com/c/WhatDoYouKnow/data. Employ Python technologies to import and inspect the raw **`training.csv`** data, isolate relevant variables, calculate new variables, organize the dataframe, and resolve missing, invalid, corrupted or duplicate values. Begin creating data visualizations to inspect variables and analyze outliers.

Import Packages, Dataset & Inspect Raw Data

After the raw training dataset was imported and saved as a dataframe, initial inspection established that there are 4,851,475 rows, or observations. Each observation holds information about a question attempted by a user. There are 17 columns, or variables. Of particular interest are **`user_id`** which identifies individual users and **`question_id`** which identifies unique questions. Since the study focuses on the performance of both, a count of each was retrieved before eliminating any records in the wrangling process. There were 179,106 users and 6,045 unique questions before data cleaning.

Organize Columns & Compute, Explore Time Data

Variables related to time were converted to datetime objects and categorical variables were converted to category type. The columns **`outcome`** and **`answer_id`** were dropped since these are not included in the test dataset. The **`question_set_id`** column was also dropped since, according to the dataset documentation provided, most question sets only have one question. The `question_id` is a better identifier. Since the original `outcome` column was dropped, the **`correct`** column was renamed to `outcome` for semantic accuracy.

Two timedate columns were created: `round_ended_at` and `round_duration`. The **`round_ended_at`** column was filled with `answered_at` values if present. If no value was found in any `answered_at` field, then the value from `deactivated_at` was used to populate `round_ended_at`. The **`round_duration`** column was created and filled by subtracting `round_started_at` values from `round_ended_at` values.

Before moving on with the dataframe, the space-separated tag strings in the `tag_string` column for each observation were converted to a comma delimited list of tag identifiers. This column was renamed **`tag_ids`**, also for semantic accuracy. Columns were then reordered to prioritize `user_id`, time data and `question_id`.

At this point there were 16 variables (columns) left in the working dataframe.

Evaluate & Resolve Missing Data

NULL values in the dataframe indicate that the value is missing, invalid or corrupted. Looking at the count of NULL values for every column in the dataframe revealed that timedate variables indeed contained missing data. NULL values were expected for **`answered_at`** since this denotes that the question timed-out. Missing values for **`date_of_test`** were also no surprise since it was known that users may or may not enter an expected test date. NULL values for **`round_started_at`** and **`deactivated_at`** were less understood and, therefore, examined more closely.

Since time data may be an important factor in improving the efficiency of testing applications, all 69 observations with missing time data for all timedate variables were eliminated. Another 21 rows were dropped for missing **`answered_at`** and **`deactivated_at`** values since `round_duration` cannot be calculated in those rows.

3,832,594 of the 4,851,475 observations in the raw dataset were missing values for **`date_of_test`**. That is nearly 80%, but this does not indicate how many users entered no test date data. A look at NULL values by user revealed that 87% of users did not provide an expected test date. This variable could not be adequately analyzed so the column was dropped.

At this point, most of the obvious NULL values in the dataframe were in the **`answered_at`** column and 10 remained in the **`deactivated_at`** column. Since observations where both are NULL were eliminated, there was no harm in keeping all of the remaining NULL values. Either variable provides the necessary information for `round_ended_at`, so ignoring one missing value is no problem.

Not at all obvious from traditional inspection of the dataframe was the fact that NULL values existed in the **`game_type`** column. This was learned from looking at the labels in the `category_labels.csv` file. Information found in the forum where the datasets are hosted claim a value of 6 represents a data collection failure. Upon inspecting observations for these values, they were dropped mainly because the round duration for each was greater than 5 hours.

Evaluate & Resolve Duplicate Data

Since one user cannot physically start or end a round at the exact same time more than once, records were inspected for such a scenario. This revealed many duplicates, but it also revealed many observations where `round_started_at` times were later than `round_ended_at` times. An inspection of such across the entire dataframe returned 357 rows. These were dropped. This prompted a look at observations where the same two variables were equals. Given the reasonable expectation that it is impossible to start and end a round at the same time, 4264 observations meeting such conditions were dropped.

(Note: time data was explored later for similar impossibilities, resulting in more dropped observations.)

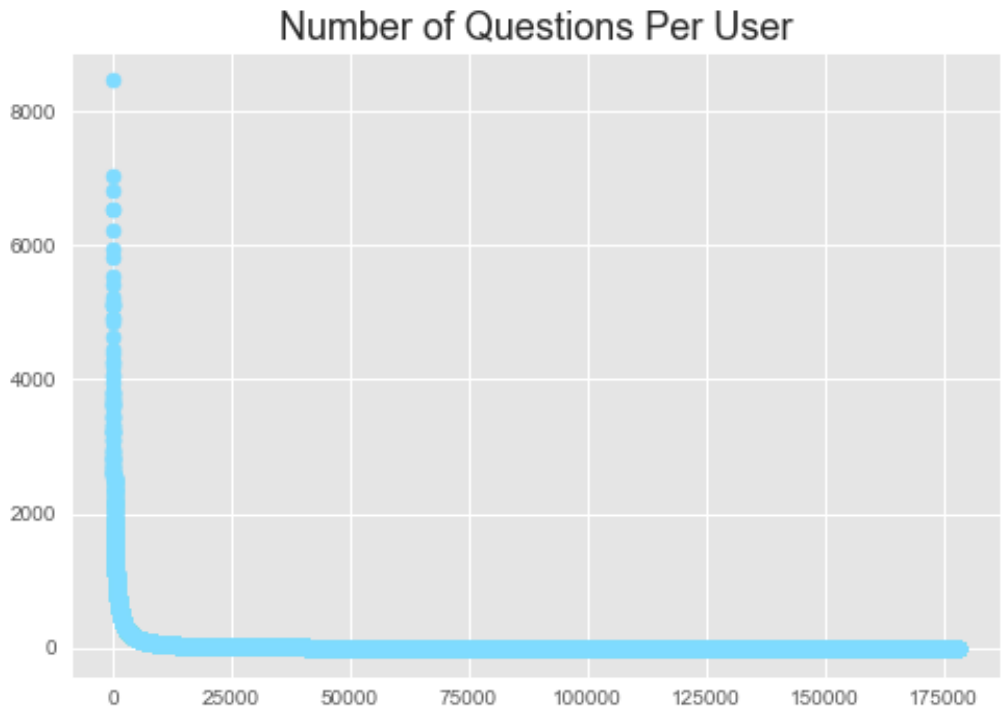
Finally, after weird time issues were resolved, duplicate observations for **`user_id`**, **`round_started_at`** and **`round_ended_at`** in combination were dropped.

At this point there were 15 variables left in the working dataframe since `date_of_test` was eliminated. Of the original 4,851,475 observations, 4,745 were dropped for unresolvable NULL values or duplicates, leaving 4,846,730 observations to explore further.

Initial Exploration

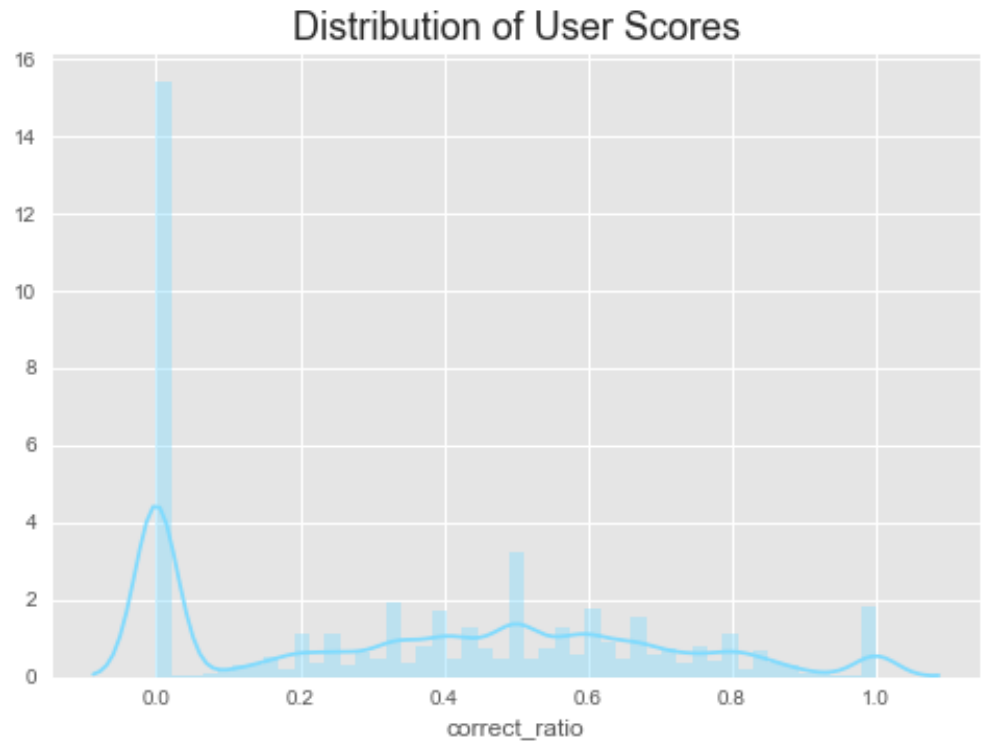
A view of time data statistics revealed that some rounds were days long and many were over within only a few seconds. 167,564 observations outside a threshold of 5 seconds to 10 minutes were eliminated. This left 4,679,166 rows in the working dataframe, for a total of 172,309 dropped records, or 3.6% of the raw dataset. This also left 178,342 users in the dataframe, so less than a half percent of the users were eliminated during data cleaning.

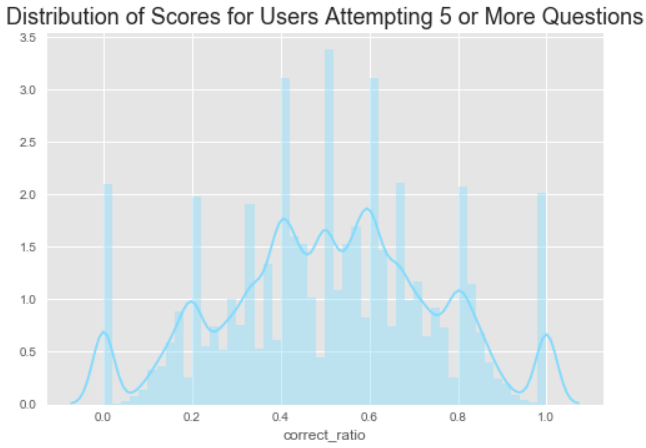
No other outliers were eliminated upon further exploration; however, it was immediately obvious that many records were skewing the data. Data visualization of questions per user, for instance, indicated that a small percent of users attempted a large number of questions and a large percent attempted a small number of questions:

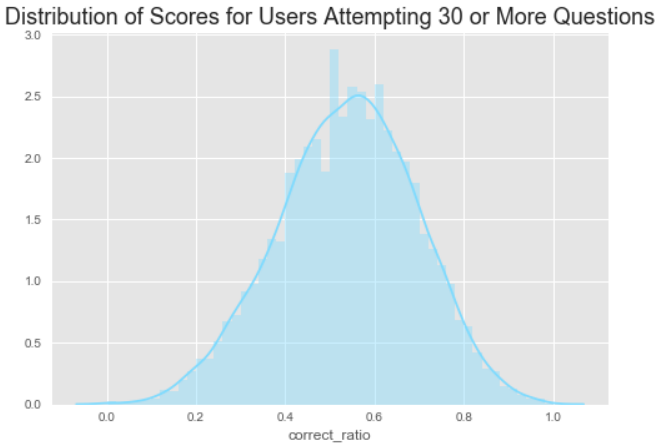


When evaluated for performance similar issues were present. These are not necessarily outliers - maybe just an effect of so much data collected, but it is also known that the training dataset includes all observations for users answering fewer than 6 questions, and that many later questions were not included in either dataset for all other users.

Rather than eliminate observations seeming to skew results, the dataframe was filtered for different user outcome views, taking into consideration that 20% of users attempted only one question, if that is ignored then the most common number of questions per user is 5, and the best scores are associated with a range of 23-38 questions:

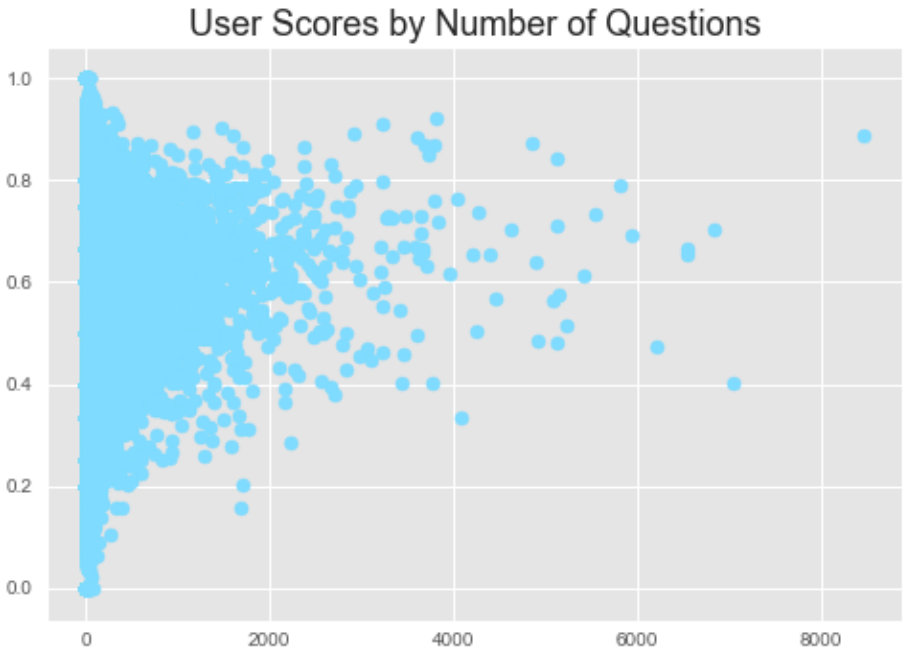






The plots definitely start to look more normal when filtered for number of questions attempted. One unexpected feature gleaned from exploration at this point was that a very high number of questions attempted is frequently associated with low scores. In fact, the average score amongst the 200 users attempting the most questions is 64%. The lowest score in that set is 28%. It would be interesting to analyze user performance over time. It might also be useful to explore whether a bot can be detected from the data.

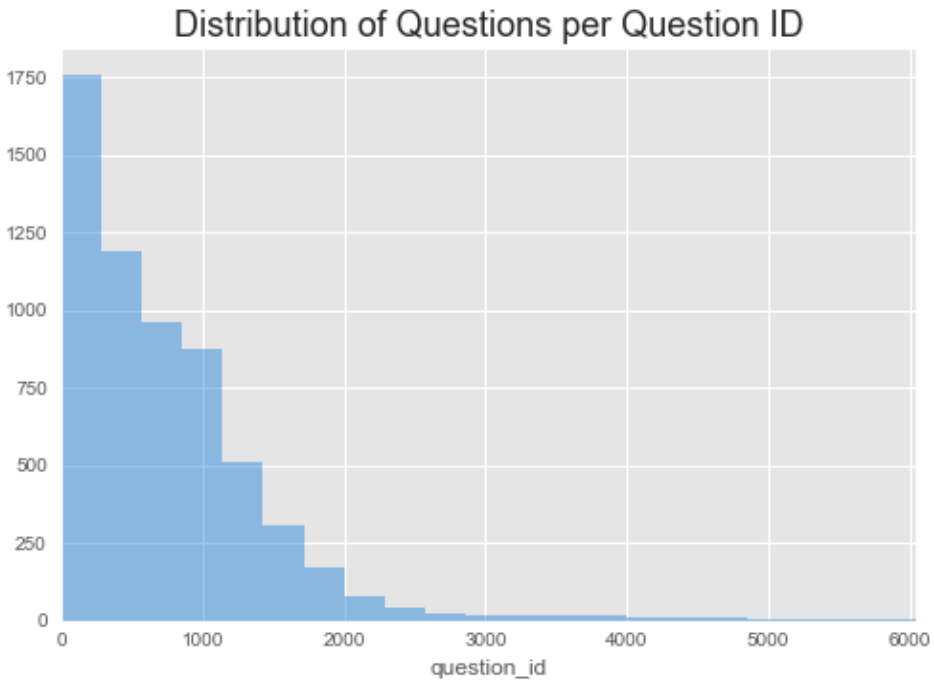
In the meantime a visualization of the user scores by number of questions was retrieved. This includes all observations in the clean dataframe:



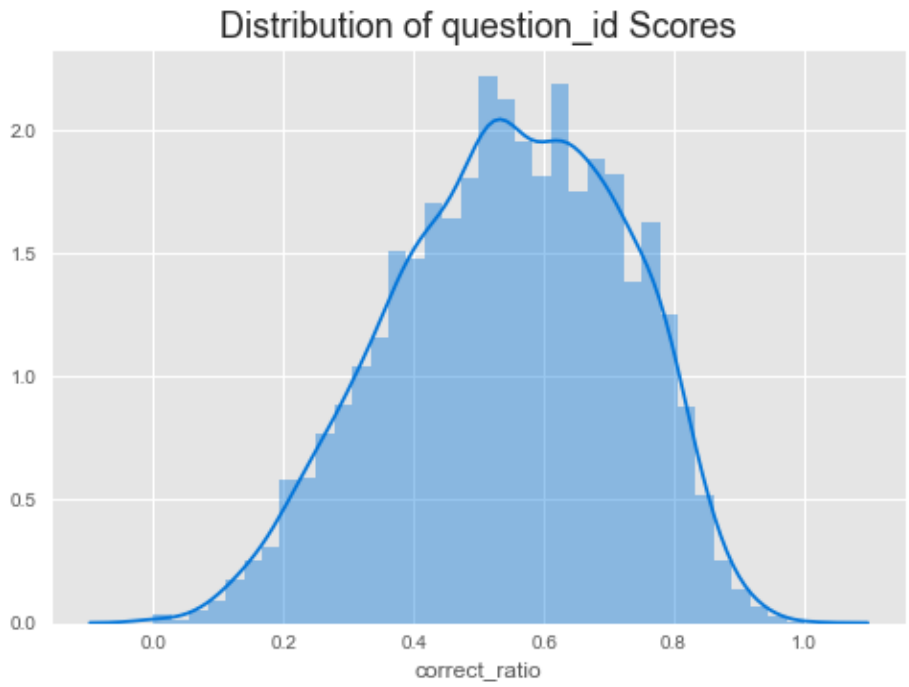
58,585 users had no correct answers, and the max number of questions attempted by those users is 72. Future analysis could also look at the distribution of questions amongst scores of zero.

The next user feature explored was the `round_duration_average`. Users were filtered for 30 or more questions and scores of 80% or higher, then sorted by time, ascending. The best `round_duration_average` for such is 8 seconds and the mean amongst the top 10 in this set is 12 seconds. The worst `round_duration_average` in the filtered dataframe is 1 minute 45 seconds and the mean amongst the bottom 10 is 1 minute 38 seconds. For those users the mean score is 82%.

The **`question_id`** variable was explored next. There are still 6,045 unique questions remaining after data cleaning. The distribution of questions per `question_id` indicates that many unique questions are underrepresented. Further analysis of other variables related to questions might explain this.



Nothing else stood out while viewing `question_id` features. The distribution of scores by `question_id` is fairly normal:



The mean score for the 10 unique questions with the best round duration average is 83%, and for those with the worst times the mean score is 84%. The dataframe was not filtered before retrieving these results, so this is for all unique questions.

Next, all other variables were inspected with these general questions in mind:

- How many are unique?
- How many questions per unique value?
- What are the statistics related to outcome?

No outliers were detected.

Given the descriptions of all **`game_type`** values, a much deeper dive into that variable should be interesting. Other than traditional practice, various games can be challenging, competitive, adaptive, user customizable, diagnostic and personalized.

One last inspection of the dataframe confirmed that there are 4,679,166 observations left for further analysis in the exploratory_data_analysis notebook.