Justin Tsao
1/12/19

Data Wrangling – Capstone Project 1

Code for this can be found here: https://github.com/jltsao88/Capstone_Project_1

- Viewing the Jupyter Notebook from the link above is the best way to follow along with this document.

Step 1 Acquire Data Source

Capstone 1 project relies on data taking from competitive ranked matches from the game League of Legends, developed by Riot Games. This was done direct through Riot Games API which can be found at https://developer.riotgames.com/ and creating a free account.

The first thing I had to do was have an account with Riot Games, which I already had from playing the game. The next step was to acquire an API key and study the API reference documentation to get exactly what I was looking for. I also noted the rate limit of 20 requests every 1 second and 100 requests every 2 minutes. As, a side note I applied for a production API which would increase my rate limit in the thousands. As of writing this document, I have not obtained a production key. The Riot Games API contains many API methods which return data in JSON. The methods I used were SUMMONER-V4 and MATCH-V4 which I will describe below.

My goal was to obtain at least 100,000 matches worth of data and store the data as CSV files to be easily turned into Pandas Data Frames. Each row of the Data Frame would be a single match and columns would each represent statistics and key information of the game such as stats of each individual player and game id. To get matches of individual games would require getting an encrypted account id based on in game player names and the individual match id every game.

The initial process to get at this data goes as follows. First, I needed to use the API method SUMMONER-V4 to get an encrypted account. I wrote a function called get_acct_id() which would take 3 arguments, the players in game name aka summoner name, region where the game is player, in this case ‘na1’, and the API key itself. I used the Python request library to return the JSON in every case. In every method I would check the response code. Code 200 meant a successful request. If the response code returned was 429 that meant I was over the limit and would have Python sleep for 2 minutes before attempting any more calls to the API. A successful call to the API return the encryted player ID

Next, I needed the match ids for 100,000 games or more. I searched well known professional players and popular streamers all who played the game at high and competitive level. All the players are considered to know the game inside and out and would lead to the most promising match data as many people tend of lower ranks tend to imitate to players in terms of game play. The API MATCH-V4 method was used to

return a list of all ranked games played in the 2018 season for each summoner name using the function get_match_history().

Next, I used the MATCH-V4 method again in a function named get_match_stats() to return the data of a match in JSON.

The main function I created to get 100,000+ match ids is called get_many_matches(). This function could take a list of in-game player names and either return a list of match ids as a new pickled file or add on to an existing pickled file. Due to the API rate limit it would currently take over 33 hour to write all the data into CSV files. I decided to split the ids into 10 files of approximately 10000 match ids in each file. This was partly due to my fear of encountering bugs and being able to monitor printed debugging outputs. I am glad I did because I caught many bugs and silent fails which led to be refining my code multiple times over and add new exception handling.

The final step was to create a function called create_master_data() that would write the both clean/wrangle the data and write the data into a CSV file. In this case multiple files because the easiest was for me to write to a csv file was to first format the data into a Data Frame. The other way would have been to write many lines of csv writing and string formatting code. Because I was converting the

Step 2 Data Wrangling and Cleaning

The way I cleaned and wrangled the data was first converting the JSON to a data frame. I saw the size in memory the data was going to use and decided to make 10 separate CSV files. As mentioned above create_master_data() would clean and wrangle the data. Many helper functions were created to achieve this. But first, I looked at the out put of the JSON. What I noticed was the champions, or the characters that could chosen to play were returned as id numbers, as were the in game items, and spells. My first step was to find a mapping of the id numbers. I found this data in JSON format by asking around on Riot Games API Discord channel found here. I was directed to other APIs which I could extract the mapping of ids to items, champs, and spells. I created a dictionary in Python for all those ids and wrote functions which would replace those id numbers with the in games. All the cleaning functions are add_player_name(), add_champ_pick(), add_sumpell_pick(), add_stats(), add_team(), and add_team_stats(). Another function create_row() would use all those cleaning functions and create a dictionary which would later be used to create rows for the function create_master_data().

The API never returned missing values which was great. However, the game is updated on almost a weekly basis meaning items are removed and changed. I ran into the case that id numbers for items and champions were missing due to different match ids occurring in different patches or updates to the game. In the case that the ids could not be match, I allowed the id numbers to stay in place rather than remove them or

create null values. I originally ran into KeyError exceptions during intital runs and did more defensive programming to my functions.

Each player had close to 100 in game stats each. Further string manipulation was needed as the stats for each player were values in key:value pairs. Therefore, I added a ‘p’ + ‘number’ before any stat name to indicate which stat belonged to each player. For example, p3 would indicate player 3. Other names for certain columns were cleaned in this fashion.

In the context of the game, there are certain outliers for stats like kills. But this is due to the nature of the game. A player can ‘snowball’ if getting first kills very early in the game. This creates a lead which other players can not keep up with. This happens from time to time. It is more common in low rank matches, but does happen in the competitive scene as well. These outliers do not need to removed at all. They are often lights of a players skill in high ranked matches. In a 5v5 game with competent players, come backs can come at any instance and even an extremely high kill to death ratio is sometimes meaningless without looking at other stats like amount of ‘vision’ is constantly being looked at on the map which can be calculated using how many vision wards are placed.

Overall I believe the data set is very reliable as I picked players in the highest ranked matches at the level of Master, Challenger, Grandmaster, and diamond. Below those tiers are platinum and gold rank being the mid level players, and the lowest being silver and bronze ranked. Of course there will be differences in the data if matches were gathered form different tiers, but I was interested in the level of play of professionals and those who stream the game online for hours at a time for a living. These are the matches that are watched and analyzed by the rest of the player base and it is usually the top ranked matches which determine the meta play of the game at any given time.