CPSC 571 Project Progress Report

Goals

Data Collection

The game we chose to collect our data on is League of Legends, an extremely popular MOBA game played all around the world. The main reason we chose this game is because of it’s accessible APIs available for data collection. By simply making an account on the website <https://developer.riotgames.com>, you can gain access to one of their API keys, which you can use to access match data, player data, or league data from any of their games. Any information that contains personal information of players is encrypted to the specific API key that you used to obtain it. These API keys also expire after about 24 hours, in which case you can register to obtain a different API key. This makes for an incredibly secure process, which we saw as a big positive for our project in terms of the ethics of our data collection. However, one negative that we ran into was the fact that we originally wanted to track individual players and compare their champion picks to the most popular champions. This proved difficult because of their encryption system, which only allowed us to track an individual player for 24 hours. While this may have worked out with an efficient script to collect data in a short amount of time, we decided to instead focus on obtaining match and champion pick data from matches at different times and different regions.

We wrote a data collection script using the python library Cassiopeia (cite). This script was set up to randomly choose one of three regions (North America, Europe West, or Brazil), as well as a tier which categorizes players into different skill levels (Diamond, Platinum, Gold, or Silver) and one of the divisions from the tier specified (I, II, III, or IV). The script then sends a request to the Riot servers to collect 200 players that meet the randomly selected categories. We had to limit the number to 200 because otherwise we would get thousands of players, and a single request could take hours. Then, one of these 200 players is randomly selected and their match history is collected. Each of these matches has around 100 different data points associated with it, so we decided to cut each match down to only the parameters we care about. This includes the matchID (unique for each match), the time it started, the region, the tier, the winning side of the match, as well as each champion selected and which team they were selected on. The script writes each match from the player selected to an output file, and then the script starts over choosing another random player. Overall, we collected 20634 unique game matches after removing a couple hundred duplicates.

Clustering

In our project proposal, we discussed using DBSCAN to cluster the data we have collected. The main reason we wanted to use DBSCAN for our specific project is because of how often Riot updates League of Legends. Our original thought was to cluster each match by the time it occurred, to ensure that every match on each cluster was played on the same or a similar patch. Since we are planning to look at champion pick rates and win rates, we wanted to ensure that matches we are classifying together have similar champion strengths, since the major thing that Riot changes in their patch cycles are champion strengths. We predicted that this form of clustering by match start time would require the clusters to be quite contained, and that would naturally create many outliers. These outliers could be detected by DBSCAN and then we could exclude them from our classification. However, in practice we found it difficult to find an epsilon value and a number of minimum neighbours that gave us more than one cluster. The three scenarios we ran into was either every point was in the same cluster, every point was an outlier, or there was one large structure with a large percentage of the data points being outliers. Since we wanted to classify our data on at least three different time slots, we decided to switch to K-Means clustering.

Our K-Means clustering was done using the SKLearn python library (cite).

Classification

Remaining Work

References

https://developer.riotgames.com/

<https://github.com/meraki-analytics/cassiopeia>

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html