CPSC 571 Project Progress Report

Goals

The main goal of our project is to create a model that accuratley predicts team performance in massively multiplayer online games. While doing research on the types of papers published in this field, we noticed a theme in the types of games that dominated these kinds of papers: MOBA games (cite). We believe the reason for this is the immense popular of these games across the globe, as well as the access to match data that the game developers have allowed the public to have. The paper we decided to use as a benchmark for our model is Kim et al. (cite). Kim et al.(cite) based their model on the proficiency-congruency dilemma. They describe this dillemma as one in which each player faces when picking a champion to play: are you more likely to win if you choose a champion that you are good at (proficiency) or one that works well with your team composition (congruency). Their research indicated that a player's proficiency with a champion is more likely to increase winning chances than team congruency. Their best model was able to accuratley predict the winning team around 60% of the time, so we decided to use this 60% number as our benchmark to beat.

While we found the model created by Kim et al. (cite) to answer an interesting question, we felt that there are better ways to accurately predict the winning team in a MOBA match. Our hypothesis is largely based on the idea of patch cycles in these games. The game we chose to collect data on, League of Legends, gets updated every few weeks. League of Legends currently has 157 different champions that players can choose to player (cite). Naturally, it is unlikely for all of these champions to have perfectly balanced power at a single point in time. Since the game gets updated every few weeks, these champions are constantly shifting in power, which creates a "meta" of the strongest champions at the time. Our hypothesis is that if we cluster league of legends match data into similar time periods, we can determine champion win rates on both the blue side of the map as well as the red side for some subset of training data. Once we have these win rates, we can use our test data to determine if the team with the higher win rate champions wins more often than not. If our hypothesis is proven correct (> 60% accuracy), then this would support the idea that a player hoping to win one of their matches should pick the champions that have the highest win rates during the current patch.

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 their website (cite), 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). We originally wanted to collect data from Korea and China, since these two countries have large playerbases and strong players compared to the rest of the world. However, the data given by the Riot APIs was in Korean or Madarin, which would have taken more work than was worth it to translate. Therefore, we stuck to only regions that used english for their data storage.

After the script has randomly chosen values for it's three parameters, it sends a request to the Riot servers to collect 200 players that meet parameters. 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 then collects each match that the player has played since June 16, 2021, which is the date that Riot added the match timestamp to the match data. The data we wanted from each match was sort, and then written to an output file. Finally, the script starts over choosing another random player, and stops only when we stop the script. Overall, we collected 20634 unique game matches between June 16, 2021 and December 2, 2021.

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 cluster 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

At our current stage, we have our data fully clustered, and a clustering algorithm which can be easily altered in case we wanted to try our model on different clusters of data. One thought we have is to also cluster the data based on region and tier. It's likely that by constricting the clusters further, we could get more accuracy in predicting the winning team using our model, since each region has their own "meta" of champions that they prefer to play. However, this would require us to create more clusters with fewer data points per cluster. It's likely that having less data to train and test on would create inconsistent results, depending on the amount of data points in each cluster. This may be a path we take depending on how well our model performs on our current clusters.

In terms of our classification, we have our Naive Bayes model determined and now all we have to do is write an algorithm to perform it. Our goal is to do 5-fold cross-validation to ensure the accuracy of our results, which is exactly what Kim et al. (cite) used to validate their model. This will be done by shuffling the data points in each cluster, and breaking them up into five different groups. We will then train our model on four of these groups, and test the model on the fifth group. This will be done five times in total, choosing a different grouping of data for each training and testing stage. The accuracy of each testing phase will be averaged, and that value will be the final result taken from that cluster. This validation ensure that the testing data we chose from our cluster was not lucky or unlucky in terms of our models ability to predict the winner.

For our final report, we will describe other papers that created models similar to ours, and compare their model directly to ours. It is our hope that the accuracy measure of our model is similar or better to what is found in the rest of the field. We will also discuss some other techniques that we would have liked to use if we had more time to learn/apply them, such as machine learning techniques.

References

Kim, J., Keegan, B. C., Park, S., & Oh, A. (2015). The Proficiency-Congruency

Dilemma: Virtual Team Design and Performance in Multiplayer Online Games

<https://developer.riotgames.com/>

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

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