Russell Feinstein

CS677 – Final Project

Due: Wed, April 27, 2022

For my final project, I have chosen to try and analyze match data from a Real Time Strategy (RTS) game I play called Heroes of the Storm. In this game, 10 players compete on teams of 5 in order to beat the other team. Suffice it to say that the goal is to win without getting into the details of how a team wins the game. I’ve chosen this as my project because RTS games generally have countless variables and decision points which we can hopefully use to create a classifier. Notably for our purposes here, this game has a cast of 90 characters, and all 10 players must use a unique character for each game. There can be no duplicate characters in a match. Given that, my goal was to see if, given just which characters were on which team, we could correctly classify the winner after training on a dataset.  
  
As such, what I’ve done here is parsed 1000 matches played by the games very best players (‘master league’) and created a dataframe where each feature was a hero and they were assigned a 1, 2, or 0 depending on if they were on team 1, on team 2, or not in the match, respectively. We choose to use a random forest, as that is typically a classifier that is good for a dataset with a large number of features, and tried to see if, after training, it could correctly classify a winner or a loser given just which heroes were in the match.  
  
Unfortunately for our purposes, it seems like the answer to that question is: no, we can not. Regardless of how many trees and what max depth was chosen for our random forest classifier, we were never able to do much better than a coinflip. Even when we were, repeated attempts to run the classifier would often result in the accuracy dropping. Only when controlling for random states could we create a contrived example where the classifier could correctly predict the winner of a game to about 55% accuracy.  
  
This is rather poor performance for a classifier. We would hope to see upwards of 80% accuracy, and ideally even higher. What this *does* mean though, is that the game is relatively balanced, which is very important for the longevity of any game in this genre (balance here refers to the idea of no hero being significantly better than any other). If our classifier was able to correctly predict winners and losers, simply by looking at which heroes were on which team, it would be safe to assume that there was some sort of reasonable imbalance. We could assume that there was some hero or combination of heroes that were good enough (or bad enough) for our classifier to confidently determine who won the game. Instead, it seems player skill and performance are the driving factors differentiating the winning and losing team.  
  
To me, while it is unfortunate for this project that we came to that conclusion, it is very interesting to see the game’s balance through this lens. Often players may complain about one hero or another being too strong, but to me, what this tells us, is that no hero (or combination of heroes) is really good enough for our classifier to be any more confident than a coinflip (give or take a margin of error).  
  
As I mentioned in my video, I was hoping to do more analysis here, but the data was simply not quite as accessible for machine learning purposes as I had hoped. Because of how much unique information needs to be stored for each player and each team, it becomes very difficult to parse. In addition, the endpoints available are much more geared towards the average user who wants to see something like a hero’s win percentage in isolation (or their win percentage on a particular hero). I’m hoping in the future, I’ll be able to work with the developer of the [www.heroesprofile.com](http://www.heroesprofile.com) website and create more API endpoints that will allow for more data analysis of the kind we are looking to do in this class and field. Something in particular (also mentioned in my recording) would be looking at which combination of upgrades have the best win rate. Currently, the website has individual upgrade choices (in isolation) win rate, but those don’t take into account the other upgrades chosen for that hero in that match.  
  
All in all, I learned through this project just how much time might get spent simply on data cleaning and wrangling. A large amount of my time on this project was spent simply accessing the data from the website and making the data readable. It is a good reminder that, while we are specifically studying Machine Learning algorithms in this class, it is important to have a wider array of Computer Science skills in order to effectively work in the field. I look forward to honing those skills and, hopefully, continuing the project I’ve started here down the line.  
  
  
No instructions necessary (I believe) to run the code. Simply run the main.py file. I’m sharing my API key in case you’d like to run the get\_and\_parse\_data.py file but that shouldn’t be necessary.