Destiny Blog Project

In many video games, the opportunity for player to receive any form of loot is tied to their ability to play well. Victory in story missions or against other players in competitive multiplayer is directly tied to getting bigger and better weapons and armor. Bungie’s Destiny hasn’t always been a shining example of this (in fact it is often the antithesis of this) after the most recent update with the release of House of Wolves, players are now more frequently rewarded for winning competitive multiplayer matches.

The point of this project is to examine trends within Destiny’s competitive multiplayer games. The idea is that if I can find a model that accurately predicts who wins and who loses, then that means there are tangible things players can do to boost their chances of winning, and therefore boost their chances for better loot.

For this project, I created a dataset by doing a random walk through some Post Game Carnage Reports about a week before the release of House of Wolves. The dataset contains over 70,000 rows, but this boils down to only being about 4,500 games worth of information. The dataset is not huge by any standards (it fits in a nice 28 MB CSV file), but it is diverse enough that I feel comfortable using it for these prediction purposes.

I used a mixture of R and Python to complete this project. I used Python with pandas to get the data and to format the dataset but used R for the actual modeling and then cleaned off the predictions in Python

# Building the Dataset

Bungie has a REST API (that they call the [Destiny Platform](http://www.bungie.net/platform/Destiny/help/)) that can be used to get Post Game Carnage Reports (and other data). A user makes a request, and the server sends them back a nice JSON file to parse. I had been working on a small Python library to handle making requests to the Destiny Platform and I had just enough of it developed to be able to use on this project. It handles requesting data and checking for errors that occur during that process.

I then built another Python script that utilizes that library to do a “random walk” through Post Game Carnage Reports. This random walk does the following: given a player’s name get their most recent game, then pick a player from that game and get their most recent game that isn’t the same as one that we already have, and then pick another player from that game and so on. I was having a problem where this fetch process would die every so often due to errors between my interface and the Destiny Platform (the source of which I still haven’t fully figured out). In order to deal with this, I created a list of starting players, and if an error occurs, the random walk jumps to the next player in the list and starts from there. This list of “anchors” includes myself and 8 other people that I know play frequently. That way, I know that their most recent game is actually recent.

This dataset is biased in that it only looks at one gametype – Control. That means that the findings from this project apply to Control, but not necessarily to all other gametypes. The reasons for picking Control are:

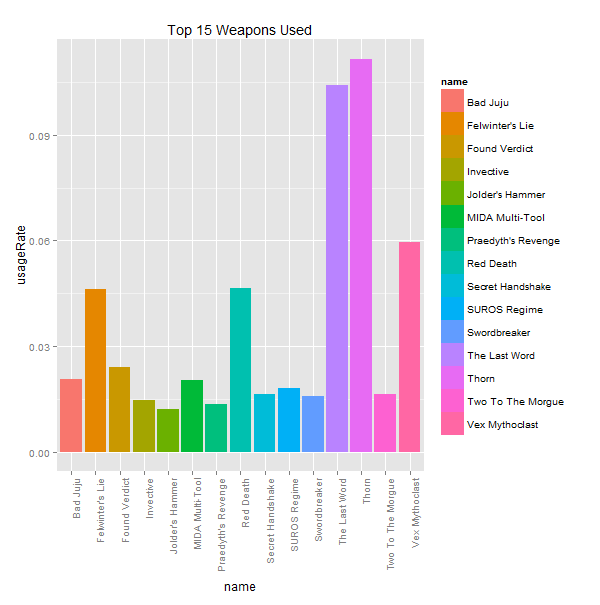
1. It seems to be more popular (or it is at the very least my favorite)
2. It’s team based
3. There are objectives other than straight kill the other player

Each of these reasons I think are significant. Since Control is the most popular, that means more people care about what how to optimize their performance for it. The fact that it’s team based adds an extra layer of complexity. Someone doesn’t need to be the “best” in order to win a game (and they can still be the best and lose a game), and our model needs to make use of features that take this into account. Objectives also throw an extra layer of dimension. In Clash, the goal the simply to kill the other team more often than they kill you. The team with the highest number of kills, and the least number of deaths, is going to win which is not necessarily true in Control. While the kills and victory are strongly correlated, high kills doesn’t always mean victory.

Every row in the dataset is a different player from a different game. There are repeat players in the dataset, but they always belong to a different game. Every column in the dataset is a different feature that I thought might be helpful in predicting who wins and who loses. The only column name which might be confusing is the “refrencedId” column, which was a typo of “referenceId” which tells you what map the match was played on.

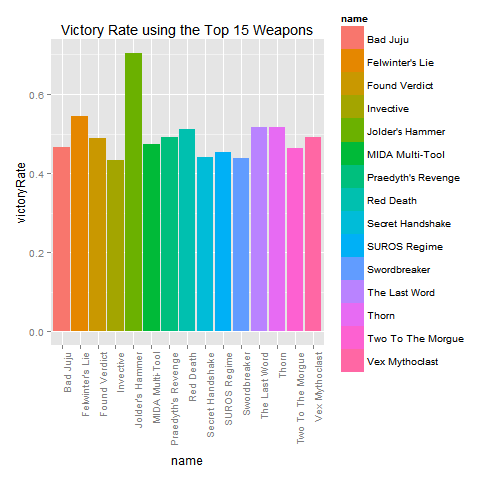
# Looking at the Data

After building the dataset, I pulled it into R to take a quick look at some trends in the data. There were a lot of them, but I am only including the ones that I found to be the most interesting.

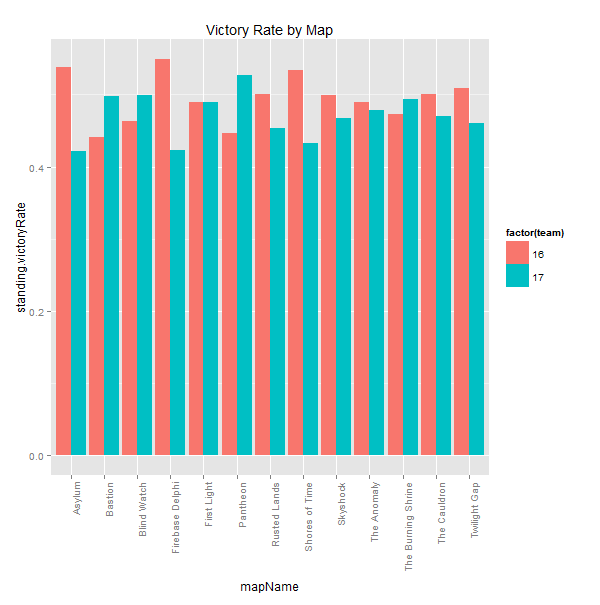


One of the first trends I went to look at was player weapon usage. From personal experience, it often feels as though the same weapons are used all the time, so I wanted to see if that was true. It turns out, that the top 15 weapons account for just over 50% of all usage. This top weapons idea is skewed and may not have been developed in the best way. While creating the original dataset, I looked at the two weapons that the player had the most kills with that game. So when I say that the top 15 weapons account, I mean that these are the 15 weapons that players individual kill the most with. Some of the top 15 most used weapons also appear in the top 15 second most used weapons. For example, Jolder’s Hammer for some people is their most used weapon at the end of a game, but is other player’s second most used weapon at the end of a game.

I then wanted to see how weapon usage is connected to victory. As you can see in the plot below, while these weapons are used very often, they don’t necessarily increase a player’s chance for victory. All of the weapons have a victory rate of about 50%, except for Jolder’s Hammer which is actually closer to 70%. Jolder’s Hammer is also the only heavy weapon to appear in the top 15. It’s uniquely high victory rate indicates to me that there might be a larger trend between the effective use of heavy weapons and victory (to be explored later).

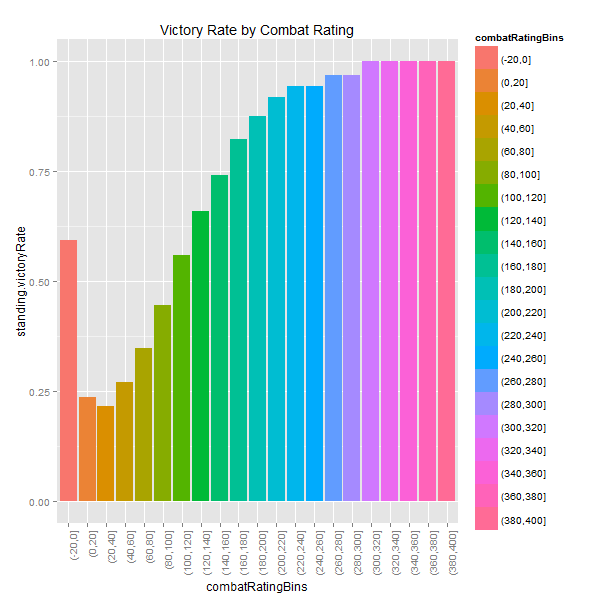


Another interesting set of factors to look at is map. It is often argued that certain teams on certain maps always win.



All things being equal, we would expect the victory rate for each team on any give map to be about 50%. According to this plot, that is not the case. Firebase Delphi is the most significant gap where Alpha Team wins 54% games versus Bravo Team who only wins 42% of their games. Note that the victory rates do not add up to 100% on any given map. That’s because each the dataset isn’t evenly split into half Alpha team and half Bravo team. When a player quits a game they are still included in the set. There are 200 extra members of Bravo team, and 1000 players who are neither Alpha nor Bravo who don’t fit in on this plot.

Another interesting factor to look at is a player’s Combat Rating. Combat Rating is a metric that Bungie created to effectively rank player’s along a scale.



Combat Rating appears to be directly tied to a player’s victory rate with the only anomaly being when a player has a combat rating of 0. Nonetheless, a player’s combat rating is a very good indicator of whether or not that player will win.

# Predicting Victory

The first step I took in predicting victory was to do a quick random forest on the data and see what happened. Doing this gave me a root mean square error of over 50%. This is unsurprising since this model attempted to predict a player’s victory based on their own stats and therefore players on the same team were being given different victory rates.

The next step was then to melt the dataset down into a team based dataset instead of a player based dataset. Each game would have two vectors associated with it – one for Alpha team and one for Bravo team. The features of each team were built off of the data of the player’s of each team.

After rebuilding the dataset in this fashion, I then split the data into training and test sets and ran 10-fold repeated cross validation on them. This gave me the probabilities that a particular team would be given a 1 for their victory variable. A value of 1 actually corresponds to defeat (this is how it is stored in the Destiny Platform) I then took this output and ran it through a Python script that looked at each game and compared these probabilities. Within each game, the winning team is the team with the smaller probability, and the loser therefore the one with the higher probability. In the event that these probabilities are the same, one team is randomly chosen to be the victor.

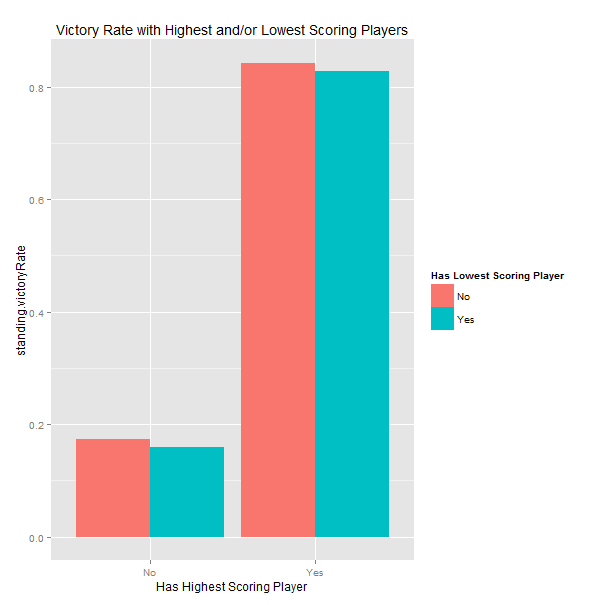
This method gave me a root mean square error of 16%. This also only used a handful of features. I was also careful to avoid features that are dead giveaways for victory (like the actual team score).

# Securing Victory

So what can players do to secure victory?

## MVP

For starters, always shoot to be the best player in the game. Seem simple and intuitive, but teams with the highest scoring player on them win over 80% of the time. This value deviates only slightly even if the team with the highest scoring player also has the lowest scoring player.



## Play with Friends

Being on a team with friends also increases chances for winning. The less fireteams on a team, thereby the more friends you being into a match with you, the more likely you are to win.

