Springboard--DSC

Capstone Project 1 - Milestone Report

By David Buehler

September, 2019

At this point in my capstone project, things have been moving right along. The question I have set out to answer is straight from the Kaggle competition website, worded as follows: "*What's the best strategy to win in PUBG? Should you sit in one spot and hide your way into victory, or do you need to be the top shot?*" The developers of PlayerUnkown's Battlegrounds (PUBG) are those interested in this question, as they are the ones that can tune the game to the way they like; and if this dataset shows that people are not playing the way the developers want them to, then they will have to come up with some ideas to change that pattern.

The data science approach that I am pursuing to address this question is to build regression models that connect the various play variables in the dataset with a variable that measures the win place percentage. Through these models I will be able to identify what variables contribute to the increase and decrease of this percentage.

This dataset was pulled from a Kaggle competition, and came pretty clean already. After some initial exploration into the data, I found that there were no missing values, and this dataset had already been broken up into training and testing data. All I needed to do was separate the solos matches from the rest of the dataset and put those into their own Pandas DataFrame. Done

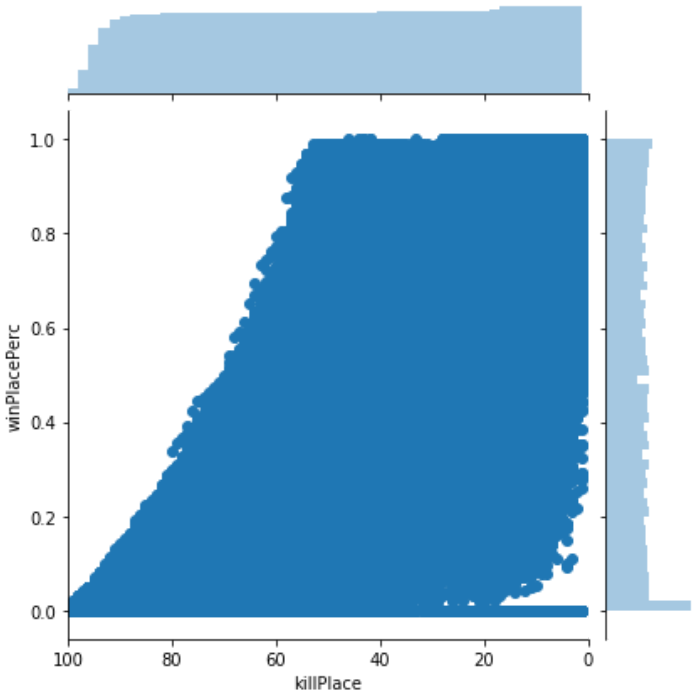
easily enough with conditional logic, pulling from the main DataFrame. Next I noticed some columns I could drop without consequence. The groupId, matchId, DBNOs, revives, and matchType columns I could drop, as DBNOs (down but no outs) and revives have no place in a solo game mode, and the other three were just categorical data that do not have any effect on our target variable, win place percentage (winPlacePerc). After doing this process to both the training and testing data, I was able to cut both files down by almost 85%. That is a huge amount and will save on memory usage and processing power in the future.

Before I dove deep into visual data analysis, I knew there was some additional cleaning of the dataset I needed to do. In early to mid 2018, PUBG had a big cheating problem. People would download programs that would make their aim in game basically perfect, not missing any shots and always getting a kill before their opponent. They would be able to do this from a very long distance away too, so the player getting shot would never even see the person shooting them, and before they knew what happened, they died. I knew I had to find these cheaters in the dataset, so I developed a criteria for cheaters. Those in the dataset with more than 50 kills were cheaters, and anyone with a headshot kill to kill ratio of 80%, as well as more than 20 kills were also cheaters. All of these people were removed from both datasets. There were not many of them, only 12 in the training data.

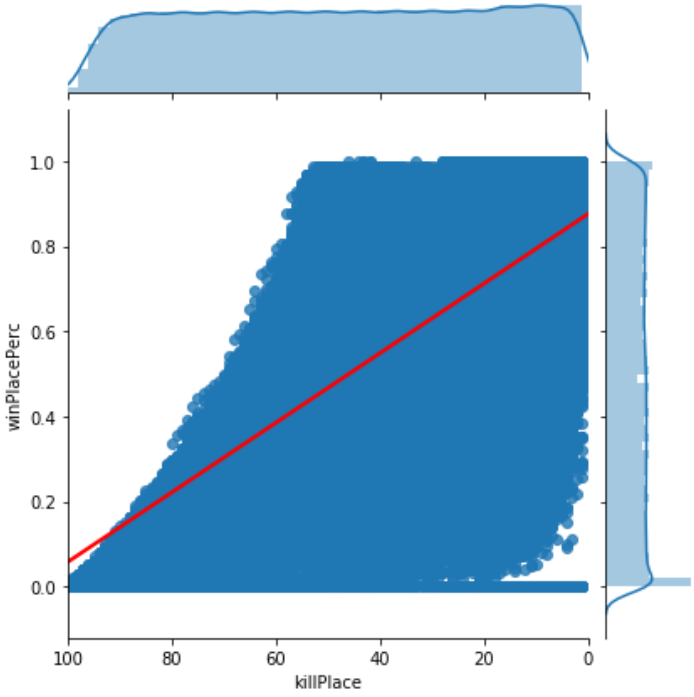
Next I took a look at match duration, and looked for matches that lasted shorter than 3 standard deviations from the mean, which in this case was about 830 seconds or about 14 minutes. The average game length in this dataset was about 1600 seconds, and speaking from

experience, with a full game of 100 people, 26 minutes is a reasonable time for a full game. However, sometimes games start with less than 100 people, sometimes as low as 1 person can start a game if everyone else in the server gets disconnected or drops out for some reason. I did this through the match duration column by dropping all those games where people may have started with only 30 people in the server. Finally I noticed that there was a maximum of two under the team kills column. In solos you can only "team kill" (kill yourself) once, so that data point needed to be dropped. Turned out there were two of them, so they were both dropped.

Next came data storytelling. This mostly consisted of exploring the effect independent variables have on win place percentage. I took a look at a number of the columns that I thought would have a strong effect on it: kills, damage dealt, kill place, heals, match duration, and weapons acquired. Out of all of these graphs, it seemed like kill place will have the largest effect on the outcome of a person's placement.



Shown above is that graph, and the linear relationship can be seen fairly easily. As a person's kill place gets better (lower is better), the more likely they are to have a better win place percentage. Here is that same graph with a linear regression line plotted:



Seen here is a clear correlation between kill place and win place percentage. Using statsmodels ordinary least squares regression (OLS), I made a linear regression model between kill place and winPlacePerc, and got an R^2 value of .576, which was the highest of the five variables tested. The next highest, surprisingly, was weapons acquired, with an R^2 value of .369. Not nearly as strong a correlation, but a noteworthy one.

Before I got into the OLS regression however, I performed a bootstrap test on the data, specifically looking at people with 5 or more kills and how they performed vs. people with less than 5 kills. I came up with a null hypothesis of: “It is as likely that someone can win a game with less than 5 kills as it is with 5 or more kills”. Performing the test, I found that people with 5 or more kills performed better than people with less than 5 kills. Those with 5 or more kills had a 95% confidence interval between 0.8941 and 0.8974, and those with less than 5 kills had a 95% confidence interval between 0.4718 and 0.4732. Just by these numbers alone we can see that the intervals have no overlap and it would be extremely unlikely to observe results like this just by chance. Performing a t-test from scipy.stats ttest_ind() function, I got a p-value of 0, thus rejecting the null hypothesis.

To finish the project I plan to use a regression approach for the machine learning portion. The target variable is a continuous distribution that is best served by using regression to model. I plan on using linear, lasso, and ridge regression for a baseline modeling approach, then random forest or k nearest neighbors, and potentially something outside of sklearn like XGBoost for a more in-depth look at the data.