Hunting Down Cheaters in PUBG

by Team ChickenDinner

Project Objective:

Cheaters are the ones who exploit game bugs to gain unfair advantages over others in games and who severely disrupt game balance. Our project's main purpose is to catch cheaters in PlayerUnknown's Battlegrounds (PUBG), one of the most popular games in the world, using anomaly detection. Our analysis is based on the assumption that cheaters are players who have absurdly above-average performance in games.

Data Cleaning and Feature Engineering:

The dataset we obtained from Kaggle consists of 4 million players' data with 29 variables.

We subsetted the data to only include matches in the normal mode, and engineered features that we believe as essential attributes to identify cheaters. First, we normalized the number of "kills" according to different number of players joined in each game. Then, we combined some similar variables such as "rideDistance", "swimDistance", "walkDistance" to "totalDistance". Moreover, we converted some variables to percentages or ratios.

Next, we incorporated the lightGBM method to understand the important features that are significantly correlated with winning the game. From the top features, we selected the attributes we believed are associated with identifying cheaters: "weaponsAcquired", "killsNorm", "roadKills", "headshotPerc", and "healsAndBoosts".

Anomaly Detection with Exploratory Data Analysis:

The first part of our EDA is about the number of kills that players made in a game. We discovered that 80% of the players only killed 0 to 2 enemies, which made a player who killed 34 people and won the game very suspicious. Therefore, we assumed that if a player kills more than 99% of other players did, this person is very likely to be a cheater. We visualized the correlation between final rank percentile and the number of kills using Plotly's boxplot.

Next, we focused on analyzing "headshotPerc", which represents how accurately a player shoots an enemy in the head. We plotted headshot percentage with final rank percentile in a displot and the graph illustrated that most players fall into 0% headshot rate and a small portion with 100% headshot rate. Skillful players could have 100% headshot rate; however, if one person kills 15 people all by headshots, it was not as simple as being lucky and skillful. Thus, we assumed that if a player kills with all headshots for more than 99% of other players do, then this one is considered to be the potential cheater.

Followed by headshot percentage, we analyzed "roadKills", the number of people a player killed by crashing over using vehicles. We found that only 0.26% of players crashed over enemies using cars, meaning killing people using vehicles is extremely difficult. Nevertheless, there's 1 player who ran over 18 enemies with vehicle, which was highly doubtful. Hence, if a player kills absurdly many enemies by vehicles, we assumed that this individual could be a cheater.

Moving around collecting weapons is common in the game. We used Plotly's bubble chart to show the correlation between weapons acquired and total distance traveled and targeted on the players who collect weapons without traveling as much. We discovered that 0.01% of players collected more than 10 weapons, just by staying where they were, without moving! One player even acquired 52 weapons without moving a single meter. We assumed this kind of "weapon-magnetic players" are possible cheaters.

Our last part of EDA was to analyze across the number of kills and the number of heals and boosts among players. We plotted the 3-D graph using Plotly to demonstrate the relationship between kills, heals and boosts, and final rank percentile. Particularly, we focused on the players who killed a lot of enemies but did not use medical supplies. This is a red flag that player might have the ability to stay alive without any help throughout the game. We observed that some players had killed 16 or 17 people all experienced 0 heals and boosts but still won the game. Therefore, we reasonably suspected players who killed over 10 people without healing or boosting to be cheaters.

In order to save computational complexity, we subsetted our data from 4 million to 0.1 million. Based on our assumptions above along with EDA, we concluded that there are 1329 potential cheaters out of 0.1 million players.

Anomaly Detection with Machine Learning Algorithms:

The last part of our project is anomaly detection using machine learning algorithms. For the data preprocessing part, we first narrowed the dataset down to 5 variables, each corresponds to one perspective that we explored during EDA. Then, we standardized the data and applied PCA on them to reduce dimensionality and multicollinearity. Through PCA, it is discovered that each component explained a significant variation of the dataset. Consequently, the dimension of the dataset was not reduced, so the PCA data was discarded. The original data was used for machine learning.

The first model built is One-Class SVM. The first step of building the anomaly detection model was to determine the proportion of anomalies in the dataset. We first treated the 1329 cheating players found during EDA to be ground truth, and then compared the proportion of these 1329 players that were caught by the 1%, 5%, 10% models, respectively. It turned out that when the anomaly proportion was set to be 10%, 98.87% of the 1329 players were labeled as cheaters by the model. This showed that 10% was a good estimation of the proportion of cheaters in the game, and thus was selected to be the anomaly proportion of the dataset. In addition, a true positive rate as high as 98.87% demonstrated that the one-class SVM model was actually identifying cheaters.

Next, another anomaly detection algorithm was implemented: Isolation Forest. This method found 91.27% of the 1329 cheating players, which also signifies that Isolation Forest was a good model to catch cheaters.

The final step is to compare and integrate the results produced by SVM and Isolation Forest models. It was discovered that 6563 players who were labeled as cheaters by both models, resulting in an overlap coefficient of 64.39%. Through this comparison process, we conclude that these 6563 players were very likely to be cheaters.

Conclusion and Future Improvement:

To sum up, we used EDA to find potential cheaters and built the anomaly detection models, one-class SVM and Isolation Forest, to test the accuracy. In the end, we found 6563 cheaters out of 0.1 million players.

Identifying cheaters has always been a huge challenge posed to the game industry. One difficulty to our models and also to the data scientists in the gaming companies is distinguishing between exceptionally good players and cheaters. One potential improvement of our project would be not only comparing horizontally across different players, but also vertically on historical data on one player. Hopefully, in this way, the cheaters can be more accurately identified.