League of Legends (LoL) 10 minutes Win Discussion

This dataset consists of the first 10 minutes of approximately 10,000 League of Legends games at a high ranked level – where the contenders are roughly of the same ability.

Source: https://www.kaggle.com/bobbyscience/league-of-legends-diamond-ranked-games-10-min/data

It is well known that the first 10 minutes of each LoL game can highly influence its outcome, but to what degree? This is what will be explored through data analysis and machine learning within this project.

Data Analysis

The dataset did not consist of any nulls and all fields were numeric – the only non-relevant information included was the gameId which allows the individual to further reference the Riot API to retrieve more information about that game. The target, "blueWins", was evenly balanced, which helped the analysis and model reach a more reliable conclusion.

Evenness of each feature

Grouping blueWins by the mean allowed me to quickly view which areas in particular skewed each game towards a win. It was found that the following statistics contributed significantly towards a win (regardless of blue or red team): firstblood, kills, assists, elitemonsters, dragons, heralds, and towers.

Interestingly, it was also found that the following made only a very slight difference towards a win: wardsplaced, minionskilled, Cspermin. Other factors, such as gold and experience also made a difference, but are already highly linked to the aforementioned features.

The following features created a significant advantage when claimed within the first 10 minutes of the game:

- 21% of all games won when that team claimed the Herald
- 49% of all games won when that team claimed the Dragon
- 60% of all games won when that team claimed FirstBlood
- An 83% win-rate if that team destroyed 1 or more towers
- 31.8% of all games being won when claiming Dragon and FirstBlood

Neural Network Models

Altogether, 4 different neural networks were used in an attempt to improve the prediction rate.

Model 1: The first model was a tensorflow model consisting of dense and dropout layers – it was a very easily trained model which gave instant results – 74% predictive accuracy.

Model 2: The basic RandomForest model achieved equivalent results to the base model, it was as quick as the base model to train also.

Model 3: A SearchGrid was used on the RandomForest model to find if any of the key hyperparameters could be altered and combined with others to improve the model. This model took significantly longer to train without any improvement on the results!

Model 4: The last model was a SupportVector model, also utilising the SearchGrid – this was quicker to train than model 3, but it still performed essentially the same as the other models at 73% accuracy.

Overall It seems that more features would need to be added for the predictive capabilities of the model to be improved; there does not seem to be even a hint, otherwise, of a way towards refining the model.