Predicting Game Outcomes Based on Player-Champion Experience in League of Legends

By: Richard Dimaria and Megh Bhavsar

Description

League of Legends (LoL) is a popular competitive computer game and the most widely played Multiplayer Online Battle Arena (MOBA) in the world. Ranked gameplay is when ten players are matched together of two teams of approximately equal skill, where each time must battle against each other to destroy the opposing team's base. At the start of each match, players will select a champion after the matchmaking algorithm forms teams. Players will often perform better on champions they have more experience on due to their differing levels of mechanical expertise on their chosen champions allowing players to make quicker and better choices which is required in the fast paced game environment. The largest impact on a player's performance is mechanical expertise and that has a similar impact on the match's outcome.

In this project we will implement a deep neural network model that can predict ranked match outcomes based on players' experience on their selected champions. The developer of League of Legends, Riot Games provides a public API with all the data needed from players and their matches.

Objective: Predict game outcomes based on a match's player-champion experience using a Deep Neural Network Model based on the model created in the paper we researched and improve the model such that it can predict the outcomes with a higher accuracy.

Sub-Objective: Implement more models to predict game outcomes based on player-champion experience such as K-nearest neighbours, Support Vector Classifiers, or Random Forest Trees then evaluate and compare their results to see which model is the most effective and why.

The proposed solution and methodology

We will query random players and filter their most recent ranked match. Using this method, we will pull a total of 5000 unique ranked matches for our dataset.

We will gather various data on each of these players to create our neural networks:

- **General Match information/stats**

- **Champion mastery points:**
  - An integer ∈ [0, ∞] approximating the player's lifetime experience on the given champion. Champion mastery points are accrued by playing matches on the champion.
- **Player-champion win rate**
  - A percentage (0.0 to 1.0) indicating the player's victory ratio for ranked games played on the given champion during the 2020 ranked season
- **Season total number of games played on the champion**
  - An integer ∈ [0, ∞] signifying the total number of ranked games the player has played on the given champion during the 2020 ranked season.
- **Number of recent games played on the champion**
  - An integer ∈ [0, 20] representing the number of ranked games that the player has played on the champion within their last 20 ranked games during the 2020 ranked season.

For each match, they also derived additional features to better reflect each team's overall player-champion experience. For both teams, they added the team's average, median, coefficient of excess kurtosis (i.e., how normal the distribution is), coefficient of skewness, standard deviation, and variance of the players' champion win rate and of their champion mastery points. The final training dataset contained 44 features per match (2 features per player for 10 players and 12 features per team).

These different features will help us predict the match outcome for a ranked game using each of these ranked games data.

Models we will use:

- k-Nearest Neighbors
  - The algorithm takes a given record *x* and compares it with the *k* closest records from its corresponding dataset, with closeness defined using some distance measure.
- Random Forest Trees
  - They used bagging to create multiple base decision tree classifiers and aggregate them into a single model. This method can yield a model with less variance and overfitting in comparison to classic decision trees. They used scikit-learn's implementation of RF trees.
- Gradient Boosting.
  - GBOOST is another ensemble method that sequentially adds predictors to its ensemble and tries to fit each new predictor based on the residual errors of the previous predictor.

        - They used scikit-learn’s implementation of GBOOST, whose default estimator is a Decision Tree.
- Deep Neural Networks
    - They utilized Keras, a Python deep learning API, to build a model. They decided to go with a pyramid network architecture, as they noticed that it performed better than networks with hidden layers with the same number of neurons in their preliminary experiments.

API Used for data: https://developer.riotgames.com/apis
- All data we will need can be found on this website

Expected outcome, deliverables, findings

It was found that player-champion win rate and champion mastery points are the only two features that had any effect on the outcome of the match.

Resources needed for project

We will use python, tensorflow, and keras, scikit learn

References

Using machine learning to predict game outcomes based on player ... (n.d.-a). https://arxiv.org/pdf/2108.02799

Work division between the team members.

Work division will be 50/50 split as we are both roommates and will be working together.