In-game winning probabilities

Idea

Real time winning probabilities given a live Ranked 5v5 solo queue game being watched in spectator mode. We actually made a proof of concept during the Riot Games hackathon in September.

Quick Start

Installing the Winning Probability app is just like any other Overwolf app. First, be sure to have a development version of Overwolf installed. Extract the archive file containing the InGameProbabilitiesApp folder. Then, open up the settings of Overwolf and navigate to the “Support” tab. In this section, click “Development Options”; this should bring up a window of all the Overwolf apps that you have installed. Click on “Load unpacked extension..” and navigate to where you extracted the “InGameProbabilitiesApp” folder and press OK with that folder selected.

To start the Winning Probabilities app, simply spectate a 5v5 game of League of Legends on Summoner’s Rift. The app should automatically start and will display the chance of each team winning in the top left and right corners of the screen. Windows Firewall may pop up the first time you start spectating. Allow any applications requested through the Firewall and you should have no issues.

Limitations

Due to not wanting to demoralize players while they are playing a game, the chances of a team winning are only shown in spectator mode. This plugin will sit and do nothing if League is started in non-spectate mode.

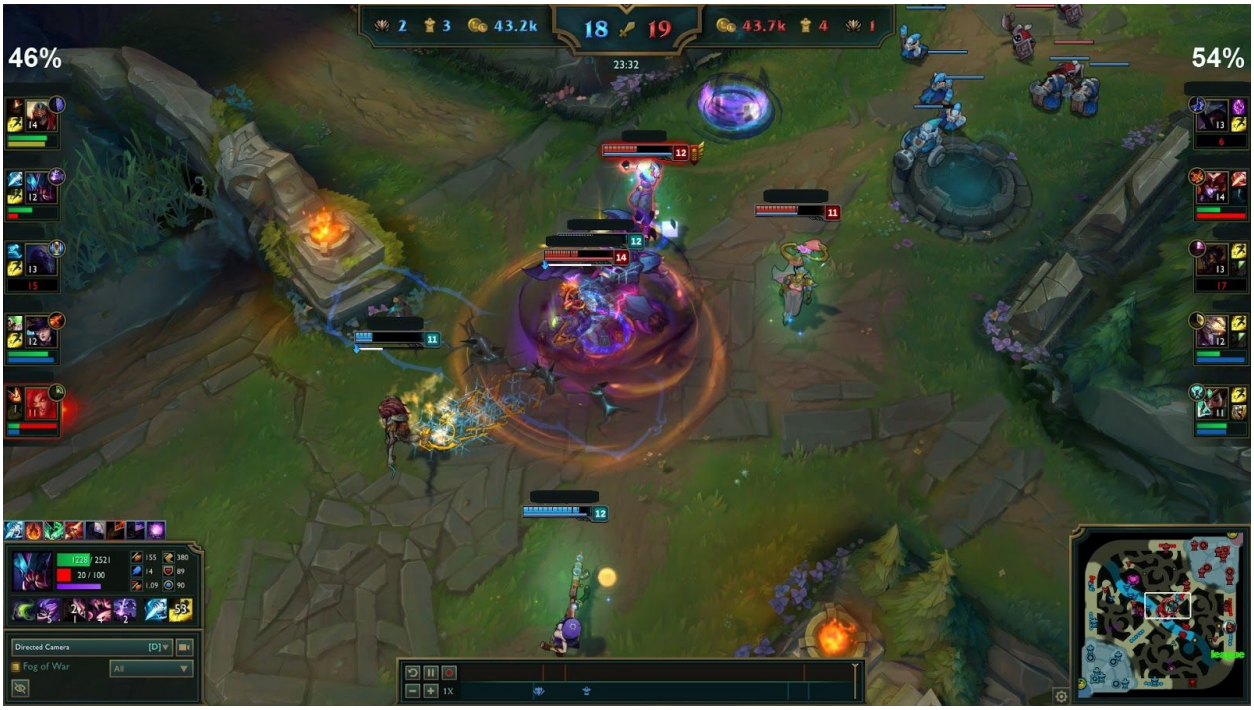
The data we have trained on is only for Ranked 5v5 games on Summoner’s Rift, so predictions for 3v3 games or ARAM games may not be accurate.

Machine learning model

We built a classification model using the random forest algorithm to predict the winning team given a game state containing information about a game at a certain point in time, such as KDA for every player, number of turrets killed per team, number of dragon killed per team, and so on.

This model was built from introspecting more than 8000 ranked games splitted in frames of one minute. This results in about 240 000 observations.

What we use are the probabilities outputted by the model. For instance, when interrogated, it will output: red team will win with 54% probability. This is what is displayed in the client.



The model was built using Apache Spark.

Overwolf App

At a high level, our Overwolf app injects a DLL into the League Client that relays all spectator UI updated to our app. The app stores this information and sends it to our Web Service to obtain the chances of either team winning. It then shows this information to the user.

In slightly more detail, we use the C# Plugin API to call into a Managed C++ assembly, which then injects the LeagueReplayHook into the League client using some WinAPI calls (VirtualAllocEx, WriteProcessMemory, and CreateRemoteThread). The league replay hook spits out all calls to a Scaleform update function out through UDP. We then listen to those UDP updates and buffer them inside of our C# plugin.

Every so often, this state information is sent to the web service to obtain a prediction of which team will win the game. When this prediction is received, an event triggers the javascript portion of our app to update the UI with the current chances of either team winning, rounded to one decimal place.

Webservice

All requests for an update of the probability of either team winning of each team goes through a webservice. Its role is simple: it relays the game data sent by the client to a game-state topic and sends back the produced prediction which was consumed from the prediction topic.

The webservice is written in node.js.

Streaming application

The streaming application reads the game-state topic containing the states produced by the webservice, interrogates our machine learning model and produces the predictions given by the model to a prediction topic.

The game state and prediction data are stored in Apache Kafka, and the streaming application is built using Apache Spark.