The software is designed in python.

Package description:

Main folder:

'temp' (an empty folder to store temporary data)

'models' (a folder containing trained models)

'Dota2WL_prediction.py' (a python file to start the software)

'trainModel.py' (a python file to train regression models)

'pergold1.csv' (a csv file containing data used for model training)

'Dota2WL_prediction.exe' (an executable file which can be directly ran, the size of this file is rather large, thus it may not be included in final submission)

About the executable file:

This file is generated by using pyinstaller. (http://www.pyinstaller.org/)

By typing 'pyinstaller --path D:/xxx/Lib/site-packages/PyQt5/Qt/bin --hidden-import sklearn.neighbors.typedefs --hidden-import sklearn.linear_model –w –F Dota2WL_prediction.py' in command line, the executable file can be automatically generated.

The reason I use this tool to generate executable file is that it does not require any configuration files, and it is really easy to install this tool.

The executable file generated has quite a large size (about 207 MB). The size of executable file generated by python file depends on libraries imported in it. As many libraries are imported this program, the size of this executable is relatively large. The size of origin python file is only 23 KB.

As the executable file does exact the same job as python file 'Dota2WL_prediction.py', the introduction will be included in following parts.

Dependencies of python file:

'csv', 'numpy', 'sklearn', 'wx', 'requests', 'json', 'os', 'shutil', 'math', 'matplotlib'.

You will need to install all these libraries to run the python files.

Details of python files:

'trainModel.py'

This file is used for training models to predict radiant team's winning rate based on radiant gold and experience advantage (can be negative if radiant team has disadvantage) on different time steps.

Running this file will generate 10 models. 5 of them is based on radiant gold advantage at time step 20%, 40%, 60%, 80% and 100% of total match time. The other 5 is based on radiant experience advantage at time step 20%, 40%, 60%, 80% and 100% of total match time.

The reason that I use different model at different time period is that, the advantage at different time has different importance on the whole match. For example, a 100-gold advantage at the beginning could be equal to 1000-advantage at the end.

The reason I use percentage of time instead of real time is that, for matches with total length 40 minutes and total length 20 minutes, advantage at 15 minutes has different importance in the final outcome. In a 40-minute match, advantage at 15 minutes may not necessarily leads to win, but in a 20-minute match, advantage at 15 minutes can greatly influence the outcome.

The models generated will be save under folder 'models' with suffix '.pkl'.

'Dota2WL_prediction.py'

This file is the user interface of the software.

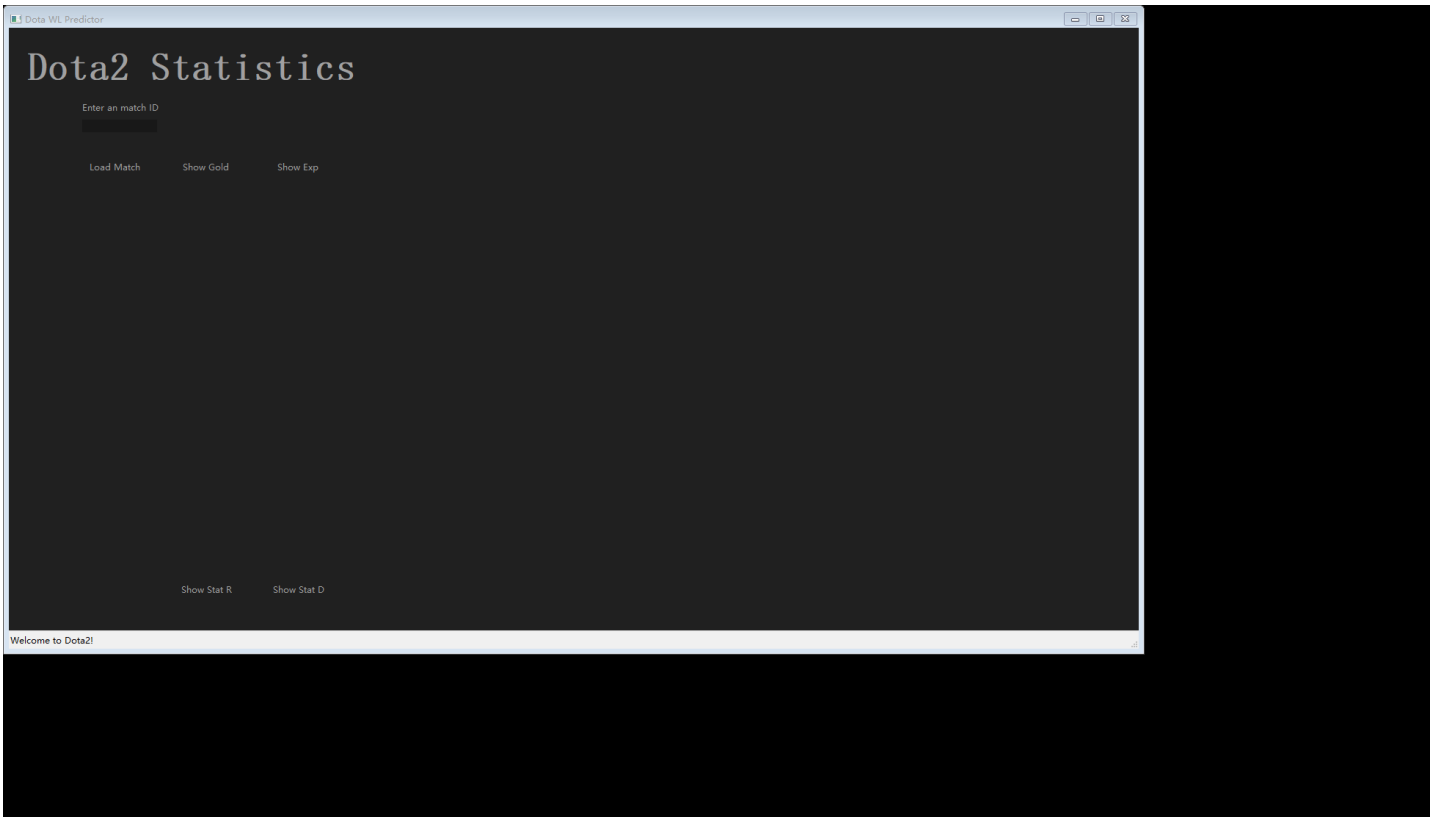
You can run this file only if you have already installed all dependencies. Any omission could cause unknown problems. Or you can run the executable file instead, which does not require any dependencies.

At the main frame ('HelloFrame'), components user can interact with are one textbox where user can enter a match ID, which is used for following actions. And there are five buttons 'Load Match', 'Show Gold', 'Show Exp', 'Show Stat R' and 'Show Stat D'.

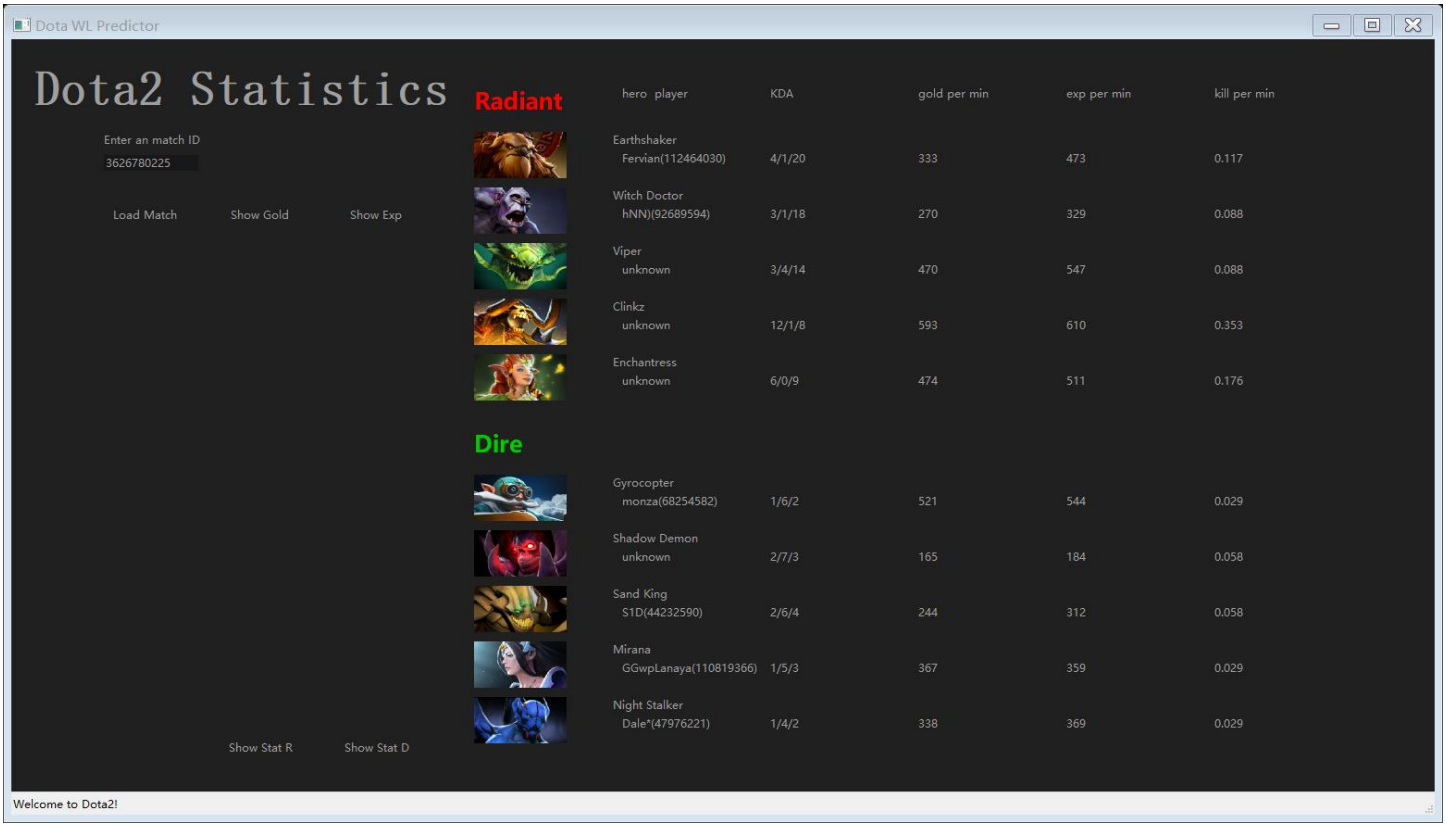
'Load Match'

This is the button you need to click first before any other actions.

You need to enter a valid match ID (you can explore matches ID on https://www.opendota.com/matches I suggest you to select a match happened at least one day ago, as some recent matches do not have all the data required in this program) in the textbox in the window first, then hit the 'Load Match' button. Once this button is clicked, the program will load match information from opendota website via apis provided by it.



Once the loading process is done, you will see something like this. Match details are listed on the right of the panel. For some players, their user names and ids are shown as ‘unknown’, this is because they do not wish to open their information to everyone, thus there is no information about them available.



‘Show Stat R’ and ‘Show Stat D’

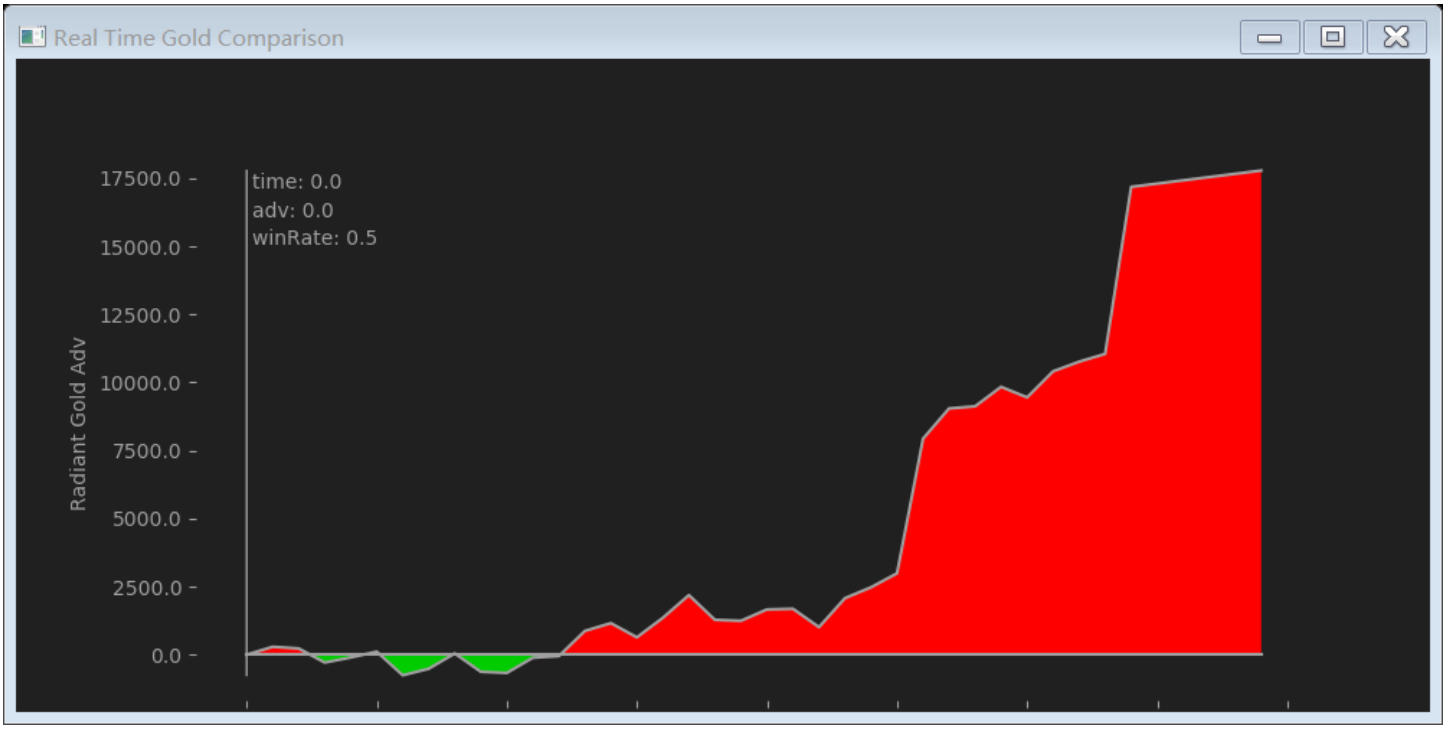
Now that the match data is already loaded, you can click any button now. Click ‘Show Stat R’ to see statistical information about radiant team. The bar graph on the top is the experience percentage in the team of each player. The red bar indicates the whole experience gain within this match of radiant team. The yellow bar indicates the experience of each player gained in this match. The bottom graph is generally the same. Only it stands for gold percentage instead on experience.

Similarly, on clicking button ‘Show Stat D’, statistical information of dire team is illustrated.

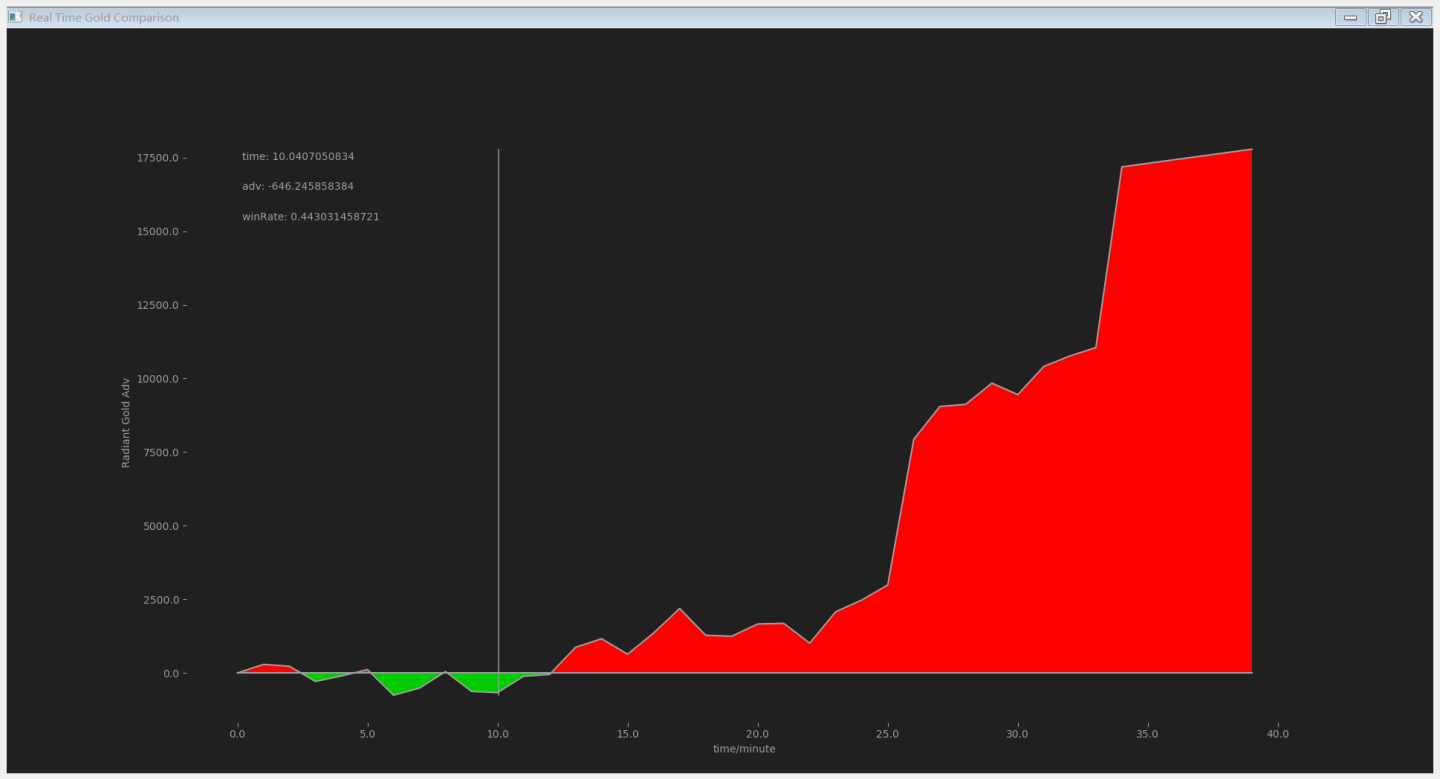
In these graphs, the larger the yellow part is, the more proportion of gold (or experience) the player takes within his team, which means the players has more importance in this team.

'Show Gold' and 'Show Experience'

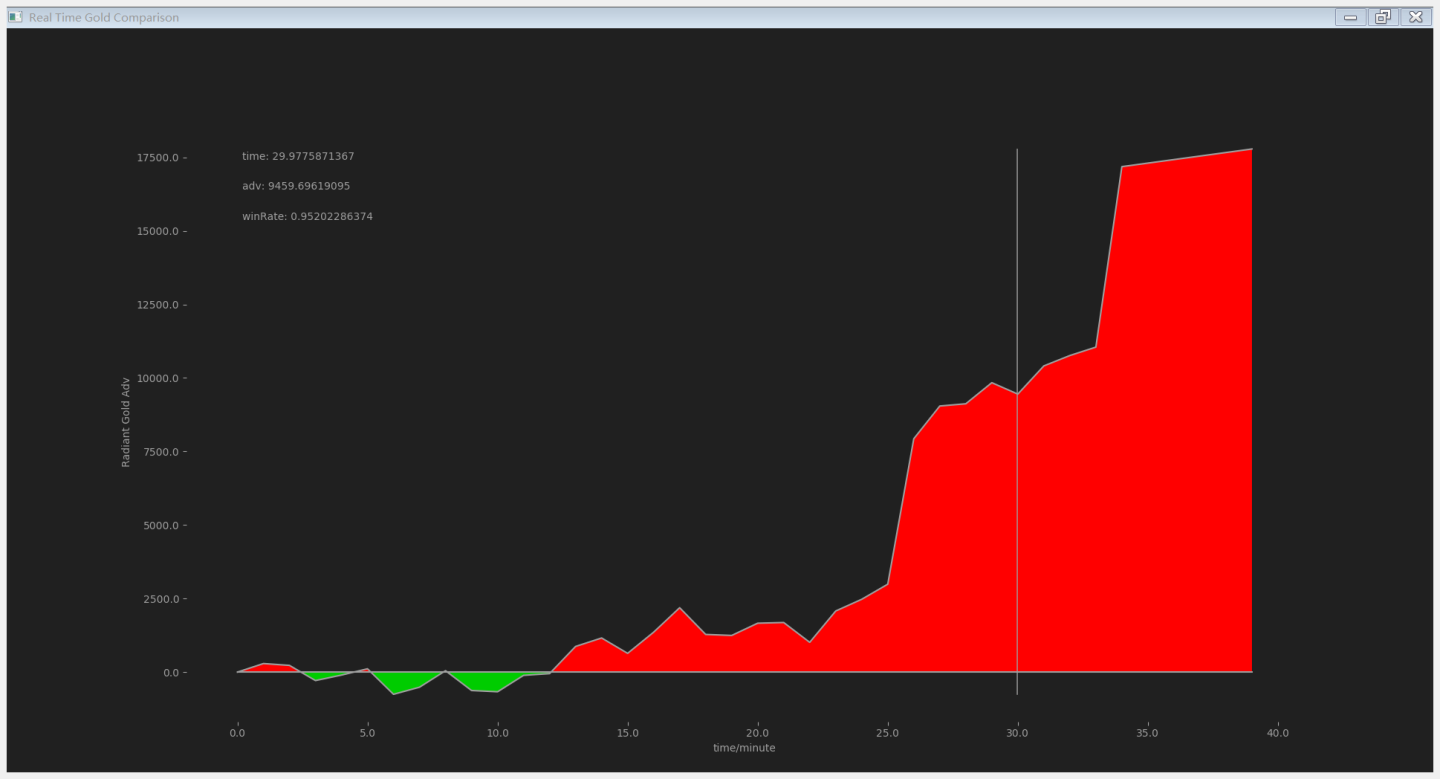
Clicking one of these two buttons will open a new frame containing a graph shows the radiant team's advantage on gold or experience. The red part is radiant team's advantage and the green part is the dire teams advantage. On the top left corner, there are information about match time ('time'), radiant team's advantage ('adv') and radiant team's winning rate ('winRate'). You can drag the cursor along the x-axis to see radiant team's advantage and winning rate at different time step.



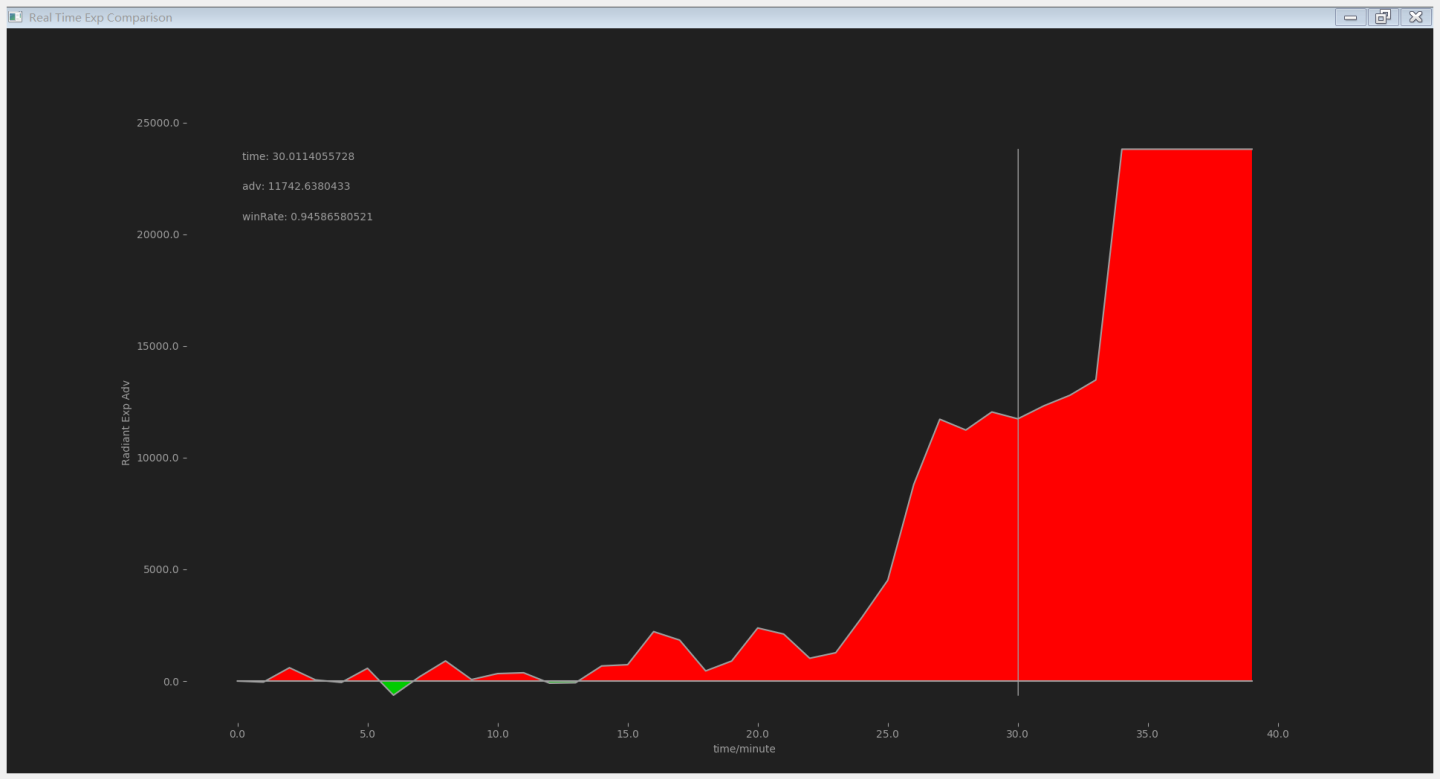
Here I will give two examples. In this graph, it shows at time 10 minutes, dire team has gold advantage 646 (shown as radiant has advantage -646). And radiant team's winning rate is relatively 0.44.



In this graph, it shows that at time 30 minutes, radiant team has gold advantage 9460 and winning rate 0.95.



This graph shows the radiant team's advantage in experience. In most cases, gold advantage and experience advantage are highly relevant. Thus, they have basically a same trend.



One thing to note here, the winning rate is calculated by 10 models trained in ‘trainModel.py’.