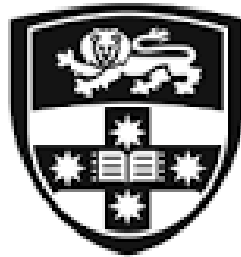


Assignment 2 Report

COMP5310 Principles of Data Science



THE UNIVERSITY OF
SYDNEY

Student name: Yifan Wang

sID: 510364904

Unikey: ywan6478

Date: 4th June 2021

DOTA2 game win rate prediction based on deep learning method

1. Abstract

The hero selection of DOTA2 is considered to be decisive for the outcome of the game. Every professional team has a coach who is responsible for selecting heroes (lineup). Because of the mutual restraint between heroes, before the game officially starts, after the heroes are selected, we can analyze the pros and cons of the lineups of both sides. We want to use deep learning to give the winning percentage of the lineup for both sides. This can help ordinary players or professional players to win the game with a higher winning lineup.

2. Background and Research problem

DOTA2(Defense Of The Ancients 2) is a free multiplayer online MOBA/strategy game developed by Valve. There are billions game matches which are played by millions of players. The data of each match can be found online, which provides us with a good data source for data analysis. Each match requires 10 players to start, and 10 players are divided into two camps, Radiant and Dire, each containing 5 players. The victory judgment of the game is very simple. The Radiant or Dire side destroys the base of the enemy camp first, and it is judged as the winner. However, it is difficult to win a game. It requires a variety of strategies and operations. The most critical strategy is the choice of game characters (usually called heroes). Before each game officially starts, each player needs to choose a hero that cannot be repeated from 112 heroes. Through a simple calculation of combinations $C(112, 10)$, there are probably more than 50 trillion possibilities for hero selection. Diversified strategic choices are the main reason why this game can attract millions of players.

For our project, we mainly study the victory predictions of Radiant and Dire after selecting 10 heroes at the beginning of each match using deep learning models. The problem can be seen as a predictive model that gives the winning percentage of Radiant and Dire after picking lineup. We use CNN, LSTM, and CNN + LSTM for the deep learning models. We will compare three models and choose one with best performance.

3. Approach Description

The dataset we selected is the dataset named "Dota 2 Matches" published on Kaggle website by Joe Ramir. It can be downloaded from the URL, <https://www.kaggle.com/devinanzelmo/dota-2-matches>. The dataset contains 50000 ranked matches from Dota2 data dump created by Opendota.

3.1. Data extraction and data cleaning

When we unzip the zip package, the size of all data files is 1.31GB, which contains 18 separate tables stored in csv file format, and a json file records the data obtained from Opendota. First, we pick out data files that

may be helpful to our project, because not all data will be used. Below we will list the selected tables one by one.

“hero_names.csv”: 112 obs. of 3 variables, contains 112 hero names and corresponding id numbers. The minimum hero id is 1, and the max hero id is 113, which id 24 is missing.

“match.csv”: 50000 obs. of 13 variables, contains main information such as start time, victory, match id, etc.

“players.csv”: 500000 obs. of 73 variables, contains the match id, ten hero id numbers selected in each game, hero damage and other game data. Because there are 10 players in each game, the number of rows in this data is ten times the number of rows in match.csv.

We use R to extract and clean data. First, we check that there are 11038020 empty (NA) values in players.csv, but there are no empty values in other two tables. Then we checked the players table carefully and found that some variables with null values are not needed by us. So, we trim the data, extract the variables(match_id, hero_id) we need to use, and detect the null values again, there is no empty values in new data table. Then we trim data from match.csv, extract match_id and radiant_win. The radiant_win contains boolean values True or False, which True means radiant win and False means dire win. We draw a bar chart of count of winning matches for both sides, showing in Figure 1. We find that radiant’s winning rate is higher than dire’s. We analyze player.csv and select the 10 heroes with the most selected and 10 heroes with the least selected. We draw a bar chart as showing in Figure 2. We found that the hero_id equal to 0 appeared 37 times, but when we checked the hero_names.csv, we found that none of the id equals to 0, the range of hero_id is from 1 to 112 represents a unique hero. When we read the documentation of the data set on the Internet, we found out that hero_id=0 means that the player is absent. So, we need to subtract these 37 missing data from the data set we are going to use in the end. Finally we plot match duration time counts over all data, and we can find there are 1853 matches’s duration time are below 15 minutes. Because there are some non-gaming factors in the high probability of the game that ends too soon, so we have to exclude these games, shows in Figure 3.

3.2. Summary statistics

After we extract and clean the data. We have a dataset with 48124 obs. and 6 variables, shows in Figure 4. We can see the match start time is all in November 2015, and the minimum time duration is 15 minutes. Radiant win has two factors: True which means radiant win, and False which means dire win. Radiant hero and Dire hero contains the id of five heroes respectively.

3.3. Data preprocessing

We use python to do data preprocessing. First, we do normalization for input data. The original input data example shows in Figure 5, we need to convert it to two-dimensional matrix whose shape is [2*113] where 2 represents the two vectors of Radiant and Dire. Each vector has 113 bits represents hero id from 1 to 113. The input data after normalization shows in Figure 6. The output part is an integer list, which represents whether the Radiant side wins, 1 for True and 0 for False. Then we use PCA do data analyzation, shows in Figure 7. As the figure shows, after PCA analyzation, we reduce to 178 bits including both radiant and dire which means 89 heroes can represent 95% of all data. We only use PCA to analyze the data, we still use the original input

data for subsequent input, because we want to analyze the lineup that includes each hero. After that, we split the samples, according to the ratio of 8:1:1, 38,500 samples as the training set, 4,812 samples as the test set, and 4,812 samples as the validation set. The role of the validation set is to observe the effect of the model on the validation set after each round of the model training on the training set. If the prediction accuracy of the model on the validation set does not improve, stop training to prevent the model from over-training the training set.

4. Evaluation Setup

Considering that the input of a sample is a two-dimensional matrix composed of two sparse vectors, here we have built a total of three models, CNN model, LSTM model and CNN + LSTM model.

For CNN model, we use a one-dimensional convolution kernel with a dimension of length 3 to convolve the input, then go through pooling and two full link operations to change the dimension to $[1 \times 1]$, and finally use the sigmoid activation function to limit the output between $[0, 1]$, that is, the winning probability of the corresponding sample. Considering that when using two-dimensional convolution, it will cross the vector, that is, the heroes of radiant and dire will be rolled together, which may not be of practical help to the prediction result. Then Conv1D is used to perform one-dimensional convolution on the input. After the convolution operation, a matrix is obtained, and then the matching MaxPooling1D () function is used to join the pooling layer.

For LSTM model, the layer directly takes the sample matrix of $[2, 113]$ as input, and generates a eigenvector of length 226. The eigenvector is dropped out and fully connected twice to become a scalar, and then the sigmoid activation function is used to limit the output between $[0, 1]$.

For CNN + LSTM model, much like the CNN model, the only difference is that the reshape operation is replaced by the LSTM layer to generate a feature vector of length 226.

Finally, we set callback function and train the models. The callback function is to check the effect of the model on the validation set after each round of training. If after this round of training, the prediction effect on the model validation set is worse than the previous round, then the callback function can adjust the learning rate or stop training operation. We use the model.fit() function to start training. tx, ty are the input and output of the training set. In the validation_data parameter, we need to pass in our validation set, and in the callback parameter, we need to pass in the callback function we set.

For tuning parameters, parameters such as the length of the convolution kernel, the dimension of the convolution output vector, and the ratio of the dropout are not fixed and can be flexibly adjusted according to the model training effect. Especially in LSTM, the hidden_size parameter is the length of the output feature vector, which is also an adjustable variable when tuning parameters.

5. Results and Analysis

We used the trained model to predict the test set. After repeated parameter adjustments, we obtained several models with good prediction effects. The highest model prediction accuracy can reach 60% for all three selected deep learning models, see Figure 8, Figure 9 and Figure 10. That is, for the task of "seeing the lineup and guessing the winner", the model can reach an accuracy of 60%. The CNN + LSTM model has the best performance over all three models, it has 2,527 matches with win rate over 60% which is larger than 2285 matches of CNN and 2222 matches of LSTM.

To have a more comprehensive understanding of the prediction effect of the model, we respectively calculate the prediction accuracy of the model on the test set, training set, and verification set, and calculate the prediction accuracy of the model when the score is high, we use the model in Figure 8 as an example. The prediction effect of the model on the training set is slightly better, exceeding 64%, while the prediction accuracy on the training set and the validation set is around 60%, and there is no particularly obvious overfitting phenomenon. In addition, of the 4,812 samples in the test, 2,285 were judged by the model to have a winning rate of more than 60%, of which 1,502 were predicted correctly, with an accuracy rate of 65.7%; 225 games were judged by the model to have a winning rate of more than 75%, of which 169 were predicted correctly with an accuracy rate of 75.1%; 64 games were judged by the model to have a winning rate of more than 80%, of which 52 games were predicted correctly, with an accuracy rate of 81.3%; there were also 6 game that the model identified as close to 85% win rate, and the prediction was correct for 4 games, with an accuracy rate of 66.7%.

To have some intuitive feeling about the prediction effect, we modify the code to let the model display the lineup of the game whose predicted win rate is greater than 85%, see Figure 11. In these 6 games, except for the first game, the model predicts the victory of dire, and all the rest predict the victory of radiant. For example, we see first match, the radiant includes 'Phantom Assassin' and 'Nature's Prophet', which are the heroes with lowest win rate, see Figure 12. This also shows that the prediction results of our model are more consistent with the conclusions displayed on the statistical level.

References:

- [1] "Dota 2 - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Dota_2. [Accessed: 18- Apr- 2021].
- [2] "Dota 2 Matches", Kaggle.com, 2021. [Online]. Available: <https://www.kaggle.com/devinanzelmo/dota-2-matches>. [Accessed: 18- Apr- 2021].
- [3] "A quick look at Dota 2 dataset", Kaggle.com, 2021. [Online]. Available: <https://www.kaggle.com/devinanzelmo/a-quick-look-at-dota-2-dataset>. [Accessed: 18- Apr- 2021].
- [4] "WebAPI/GetMatchDetails - Official TF2 Wiki | Official Team Fortress Wiki", Wiki.teamfortress.com, 2021. [Online]. Available: https://wiki.teamfortress.com/wiki/WebAPI/GetMatchDetails#Tower_Status. [Accessed: 18- Apr- 2021].
- [5] "Epoch Converter", Epochconverter.com, 2021. [Online]. Available: <https://www.epochconverter.com/>. [Accessed: 18- Apr- 2021].
- [6] "Portal:Patches - Liquipedia Dota 2 Wiki", Liquipedia.net, 2021. [Online]. Available: <https://liquipedia.net/dota2/Portal:Patches>. [Accessed: 18- Apr- 2021].
- [7] "OpenDota - Dota 2 Statistics", Opendota.com, 2021. [Online]. Available: <https://www.opendota.com/>. [Accessed: 18- Apr- 2021].

Appendix:

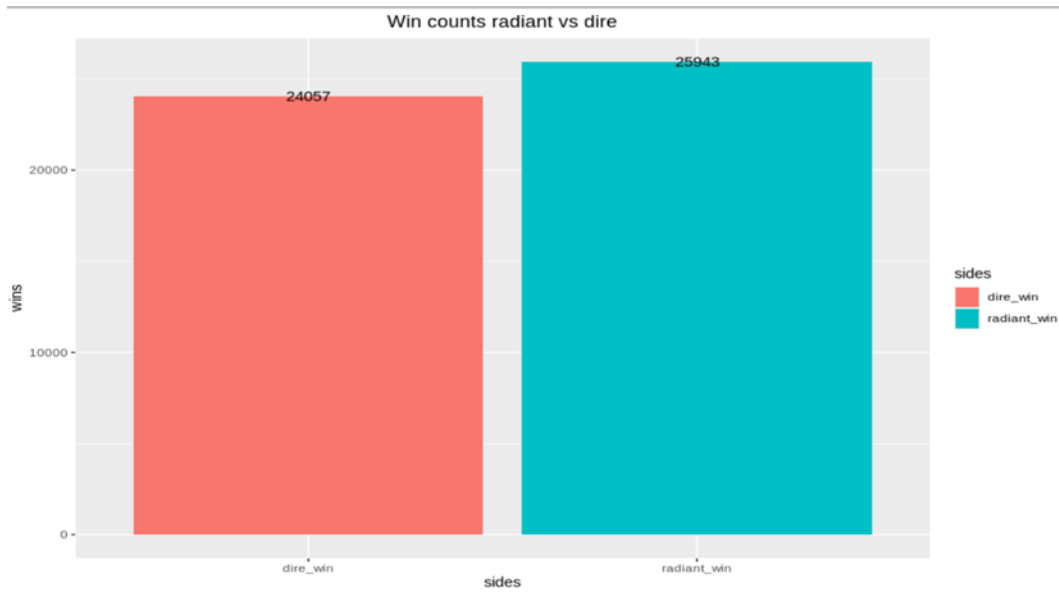


Figure 1. Win counts radiant vs dire

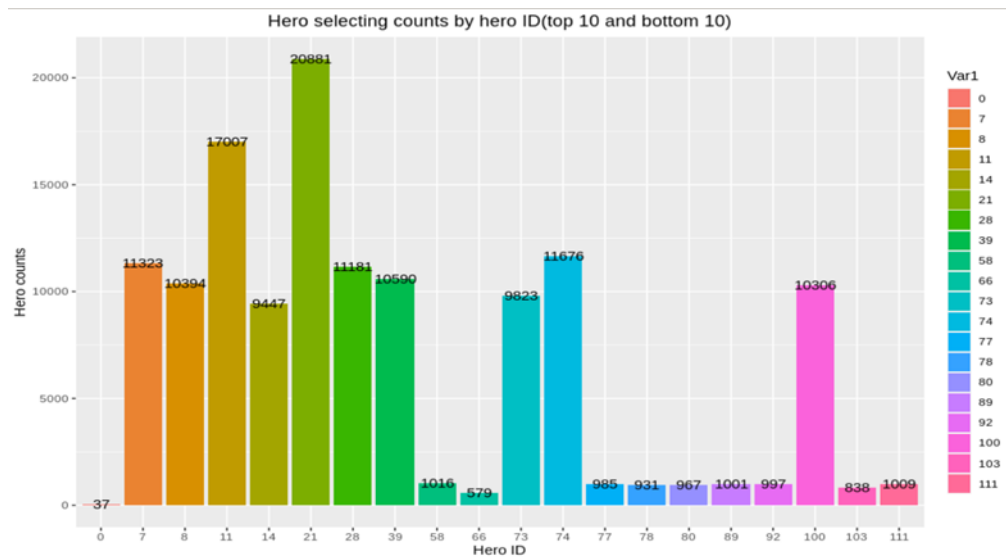


Figure 2. Hero selecting counts (top 10 and bottom 10)

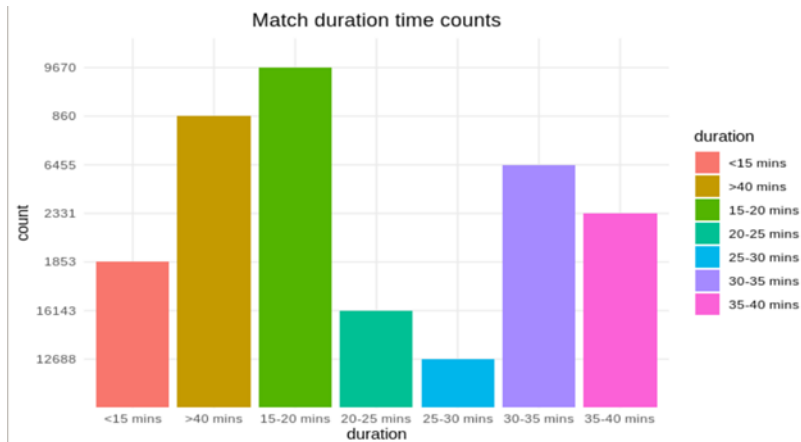


Figure 3. Match duration time counts

match_id	start_time	duration	radiant_win
Min. : 0	Min. :2015-11-05 19:01:52	Min. : 1500	False:23373
1st Qu.:12543	1st Qu.:2015-11-13 23:17:53	1st Qu.: 2072	True :24751
Median :25024	Median :2015-11-15 08:50:50	Median : 2443	
Mean :25020	Mean :2015-11-15 07:41:45	Mean : 2523	
3rd Qu.:37527	3rd Qu.:2015-11-16 19:04:12	3rd Qu.: 2890	
Max. :49999	Max. :2015-11-18 06:46:55	Max. :16037	
radiant_hero_id	dire_hero_id		
Length:48124	Length:48124		
Class :character	Class :character		
Mode :character	Mode :character		

Figure 4. Data statistics

```
radiant hero id: 86,51,83,11,67
dire hero id: 106,102,46,7,73
```

Figure 5. Input data example

```
sample_in shape: (48124, 2, 113)
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Figure 6. Input data after normalization example

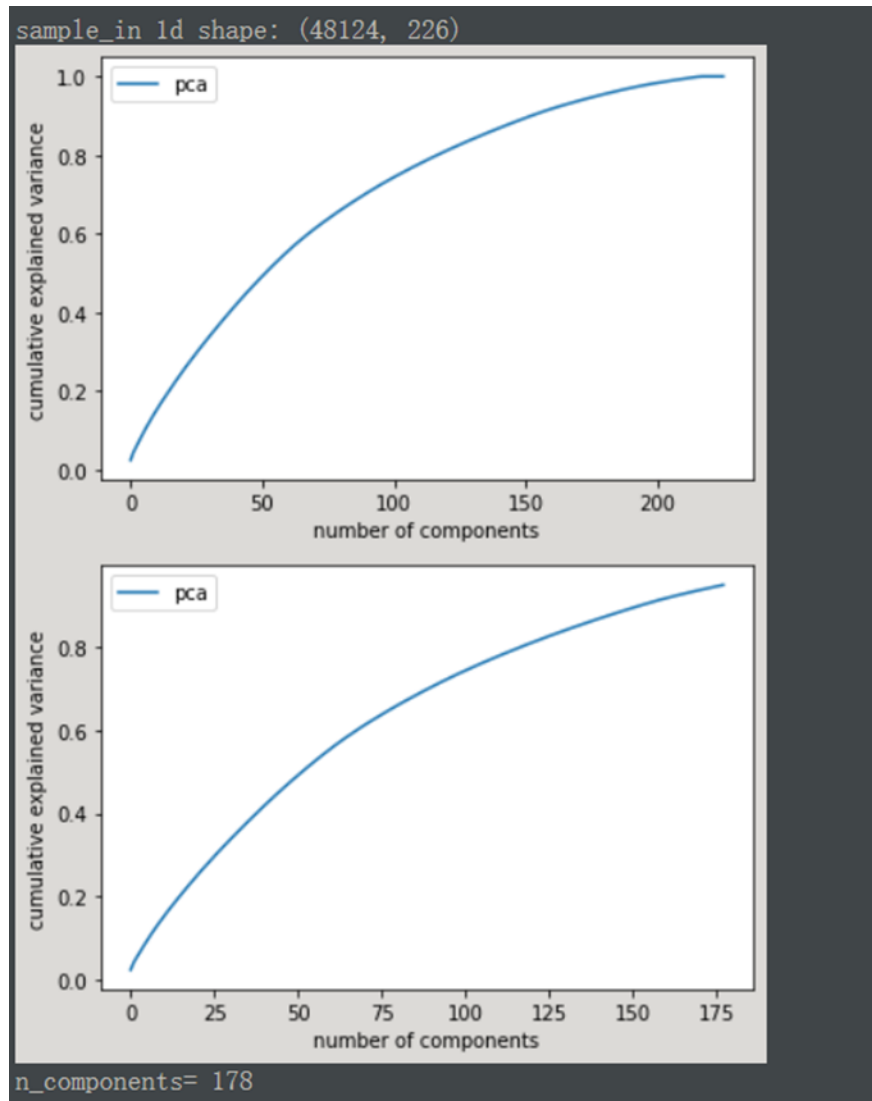


Figure 7. PCA for input data

3.1 CNN

```

1 model_name = '5310_cnn'
2 test_model(model_name)
3 print('\n')
4 get_hero_id(model_name, 0.85)

```

Test dataset accuracy: 0.594139650872818
 Train dataset accuracy: 0.6426493506493507
 Validate dataset accuracy: 0.5987115544472152
 Test dataset, the accuracy of the predicted win rate over 0.6: 0.6573304157549235 (1502/2285)
 Test dataset, the accuracy of the predicted win rate over 0.75: 0.7511111111111111 (169/225)
 Test dataset, the accuracy of the predicted win rate over 0.8: 0.8125 (52/64)
 Test dataset, the accuracy of the predicted win rate over 0.85: 0.6666666666666666 (4/6)
 Test dataset, the accuracy of the predicted win rate over 0.9: 0.0 (0/0)
 Test dataset, the accuracy of the predicted win rate over 0.95: 0.0 (0/0)

Figure 8. CNN model accuracy

3.1 LSTM

```
1 model_name = '5310_lstm'  
2 test_model(model_name)  
3 print('\n')  
4 get_hero_id(model_name, 0.85)
```

```
Test dataset accuracy: 0.6012053200332502  
Train dataset accuracy: 0.6124155844155844  
Validate dataset accuracy: 0.6014131338320865  
Test dataset, the accuracy of the predicted win rate over 0.6: 0.6633663366336634 (1474/2222)  
Test dataset, the accuracy of the predicted win rate over 0.75: 0.7591623036649214 (145/191)  
Test dataset, the accuracy of the predicted win rate over 0.8: 0.7441860465116279 (32/43)  
Test dataset, the accuracy of the predicted win rate over 0.85: 0.5 (2/4)  
Test dataset, the accuracy of the predicted win rate over 0.9: 0.0 (0/0)  
Test dataset, the accuracy of the predicted win rate over 0.95: 0.0 (0/0)
```

Figure 9. LSTM model accuracy

3.1 CNN + LSTM

```
1 model_name = '5310_cnn_lstm'  
2 test_model(model_name)  
3 print('\n')  
4 get_hero_id(model_name, 0.9)
```

```
Test dataset accuracy: 0.5978802992518704  
Train dataset accuracy: 0.638961038961039  
Validate dataset accuracy: 0.5999584372402328  
Test dataset, the accuracy of the predicted win rate over 0.6: 0.6553225168183617 (1656/2527)  
Test dataset, the accuracy of the predicted win rate over 0.75: 0.7184986595174263 (268/373)  
Test dataset, the accuracy of the predicted win rate over 0.8: 0.7154471544715447 (88/123)  
Test dataset, the accuracy of the predicted win rate over 0.85: 0.625 (15/24)  
Test dataset, the accuracy of the predicted win rate over 0.9: 1.0 (1/1)  
Test dataset, the accuracy of the predicted win rate over 0.95: 0.0 (0/0)
```

Figure 10. CNN + LSTM model accuracy

```

Print the line-up with win rate over 0.85

radiant heros: ['Windranger', 'Phantom Assassin', 'Nature's Prophet', 'Broodmother', 'Rubick']
dire heros: ['Crystal Maiden', 'Earthshaker', 'Shadow Fiend', 'Venomancer', 'Ursa']
predicted win rate: 0.13141668
actual win rate: [0.]

radiant heros: ['Pudge', 'Witch Doctor', 'Spectre', 'Silencer', 'Undying']
dire heros: ['Phantom Assassin', 'Rubick', 'Visage', 'Magnus', 'Ember Spirit']
predicted win rate: 0.8746617
actual win rate: [1.]

radiant heros: ['Venomancer', 'Spirit Breaker', 'Silencer', 'Tusk', 'Abaddon']
dire heros: ['Anti-Mage', 'Sniper', 'Queen of Pain', 'Huskar', 'Legion Commander']
predicted win rate: 0.85363185
actual win rate: [1.]

radiant heros: ['Lich', 'Necrophos', 'Night Stalker', 'Spectre', 'Spirit Breaker']
dire heros: ['Bloodseeker', 'Queen of Pain', 'Bristleback', 'Ember Spirit', 'Earth Spirit']
predicted win rate: 0.8687806
actual win rate: [0.]

radiant heros: ['Pudge', 'Lich', 'Riki', 'Doom', 'Legion Commander']
dire heros: ['Phantom Lancer', 'Chen', 'Ancient Apparition', 'Bristleback', 'Ember Spirit']
predicted win rate: 0.86114883
actual win rate: [1.]

radiant heros: ['Earthshaker', 'Zeus', 'Spectre', 'Outworld Devourer', 'Undying']
dire heros: ['Shadow Fiend', 'Windranger', 'Nature's Prophet', 'Legion Commander', 'Ember Spirit']
predicted win rate: 0.859676
actual win rate: [0.]

```

Figure 11. Line-up with win rate over 0.85

















Hero										
	Pick %	Win %	Pick %	Win %	Pick %	Win %	Pick %	Win %	Pick %	Win %
 Phantom Assassin	18.54%	46.58%	15.55%	45.56%	13.76%	45.48%	12.21%	45.69%	10.41%	44.58%
 Nature's Prophet	7.42%	41.80%	5.15%	40.18%	4.25%	41.31%	3.74%	42.98%	3.85%	44.87%
 Enchantress	3.52%	45.85%	2.77%	44.75%	2.58%	45.06%	2.99%	45.13%	4.99%	45.28%
 Shadow Fiend	18.66%	47.24%	17.86%	46.60%	18.42%	46.56%	18.19%	46.17%	13.94%	45.52%
 Troll Warlord	7.24%	49.64%	5.44%	48.06%	4.61%	46.43%	4.35%	46.18%	4.29%	45.98%
 Rubick	11.55%	44.88%	13.98%	44.76%	15.55%	45.68%	16.04%	46.43%	15.49%	46.61%
 Death Prophet	3.17%	45.78%	2.80%	46.01%	2.47%	47.03%	2.44%	46.33%	2.83%	46.67%
 Pudge	26.89%	50.59%	26.59%	49.47%	26.30%	48.79%	25.27%	48.39%	19.47%	46.71%

Figure 12. Lowest win rate hero in ranked match