Predicting Outcome of DOTA2 Games

STAT 432 Final Report

Will Jeziorski Zhiyuan Xie and Julian Nieto

May 1, 2019

Contents

Introduction

Before we dive into the paper, we want to provide the reader with a link to our data cleaning rmd file, as it is rather large and we don't want to include it in our paper for space considerations. We also provide the reader with two more links in the repository which contain both the training and testing data set.

Link for Data Cleaning File:
https://github.com/JulianFN/Stat432Project/blob/master/Project%20Data%20Cleaning%20File.Rmd

Link for Training Data: https://github.com/JulianFN/Stat432Project/blob/master/train.csv

Link for Testing Data: https://github.com/JulianFN/Stat432Project/blob/master/test.csv

Game description

Dota 2 has been one of the most famous multiplayer online battle arena games since 2013. The game has five people play against five other people on two different teams called Radiant and Dire. Player pick heroes, and each hero has unique abilities. The team that wins does so by destroying the enemy base, called the ancient.

Our goal

For our project, our goal is to find which factors will be important in determining the winner. To do this, we needed to create a data set that allowed us to perform an analysis on which team won for a single game. After creating this data set, we want to perform various binary classification techniques such as neural networks and logistic regression.

Data Description

Our dataset comes from OpenDota.com, and in our project, we specifically use their YASP dataset for doing the analysis. The aggregate dataset contains approximately five- hundred games in 2015, and each game have data of variables, some of which are shown below.

- Side of Winner
- Tower status
- Number of Chats
- Structure Status
- Creeps Score

This data was initially contained in a JSON file that was about 100 GB large. We then used the previously mentioned data cleaning file to clean the data and then dropped the first column of the data (as it was just X values.)

The link to the source of the data set we used is in the references section of this paper.

Literature Review

A team coming from University of California, San Diego has done the a similar analysis based on logistic regression and random forest classifier. They first used post-game statistics, which included gold per minute, XP per minute, and kill per minute. These data are considerably important because the amount of gold can determine which items the player could buy, and XP could show hero's level status. Using this dataset, their logistic regression and random forest gave them a win rate around 99%. On the other hand, they use only heroes ID from from each side and synergy for each two heroes from each side to construct models, which give them a lower rate about 70%.

Summary Statistics and Data Visualization

Data Cleaning

For this project, we needed to access thousands of games worth of DOTA 2 data. To do this, we first downloaded about Open DOTA website. This website provides information about all aspects of the game, ranging from the macro-game values such as who won the game and what objectives were taken to the micro-game data involving individual creep score and lane position. As a result, one observation contains a lot of data. The first observation of one of our initial data sets was a 14 minute and 58 second game. When we extracted the observation into a list in R we found that it was 145 pages of text in Microsoft Word if we allowed 3 columns per page of observations.

Clearly, we cannot do analysis on a data set this large. As a result, we decided to extract only the variables we thought would influence a win for either side. When we were cleaning the data, we ran into a couple of issues. First, we ran into an issue where there could be games that don't initialize and don't end with a winner. Luckily, this caused our function to fail to run, and we were able to drop the putliner. Clearly, there are more variables that we could extract, but these are the ones we decided to analyze given our time constraints. As a result of this cleaning, we shrunk our about 11 gigabytes of data to two files totaling 27 megabytes of data.

Here is a quick sample of our data. Our training data set and testing data set have the same format. For our training data set, we have 50000 observations, while our testing data set has 10000 observations so we have an 83%/17% split of our data. The data output below is the first 6 observations of the training data set and only the first 15 columns are output from this sample.

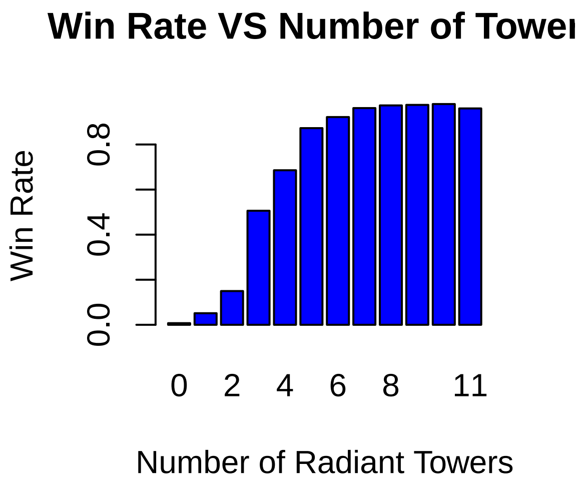
```r
head(train_data,6)[,1:15]
```

```
##   RadiantWin TimesChatted RadiantTowerStatus DireTowerStatus
## 1          1            3                 11              10
## 2          1            3                 10               1
## 3          0            0                  0              11
## 4          1            2                 11               1
## 5          1            7                 11               3
## 6          0           85                  3              11
##   RadiantBarracksStatus DireBarracksStatus RadiantFB Hero1 Hero2 Hero3
## 1                     6                  6         0     0     0     0
## 2                     6                  2         1     0     0     0
## 3                     0                  6         0    -1     0     0
## 4                     6                  2         1     0     0     0
## 5                     6                  4         1     1     0     0
## 6                     2                  6         0     0    -1     0
##   Hero4 Hero5 Hero6 Hero7 Hero8
## 1     0     0     0     0     0
## 2     0     0     0     0    -1
## 3     0     0    -1     0     1
## 4     0     1     0     0     1
## 5     0     0     0     0     0
## 6     0     0     0     0     0
```

To get a sense of what are data involves, we created a few charts to familiarize the reader with some of the things our data set holds.

First, we created a bar plot visualizing the win rate vs number of towers. This graph will help the reader see how towers affect the winrate.

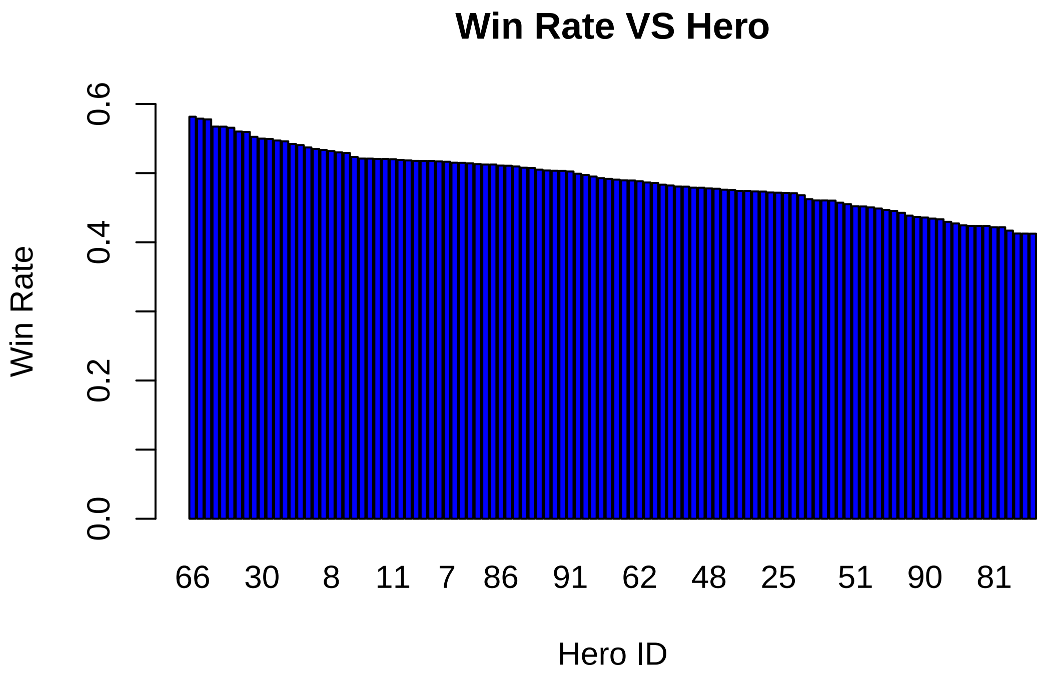
```r
barplot(accuracy, names.arg = M, xlab="Number of Radiant Towers",ylab="Win Rate",col="blue",
main="Win Rate VS Number of Towers")
```



In general, we see that win the radiant has more towers they end up winning more. This makes perfect sense, as towers prevent the enemy team from running into the base and destroying it.

Next, we created a bar plot showing how win rate differed between heroes.

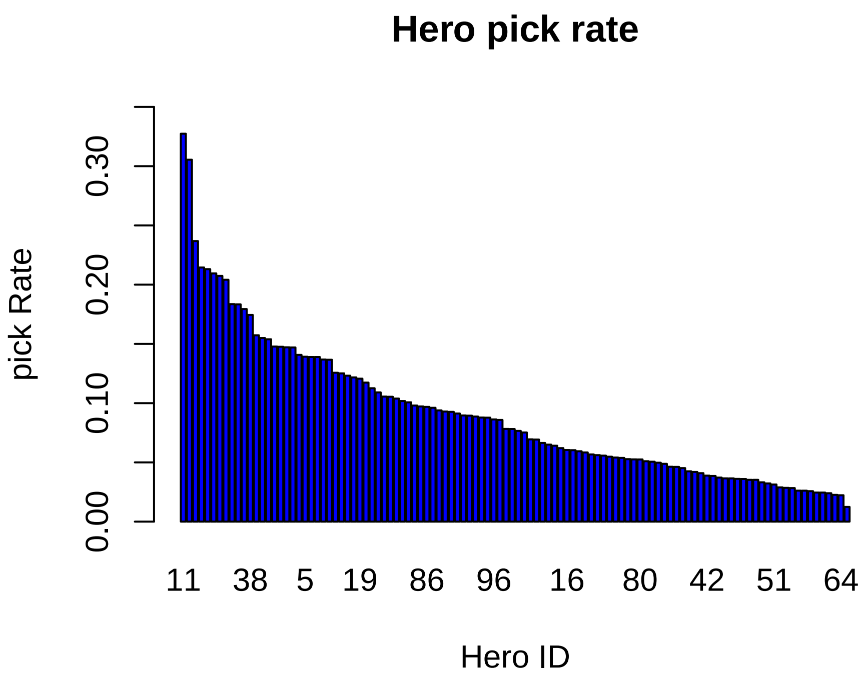
```r
barplot(sort(accuracy, decreasing = TRUE), ylim = c(0, .60), xlab="Hero ID",ylab="Win Rate",col="blue",
main="Win Rate VS Hero")
```



From this plot, we identified that Chen, Clinkz, and Skywrath mage all had the highest winrate with about 58%, while Storm Spirit, Puck, and Night Stalker had the lowest winrates each with about 41%. Having played some DoTa, Storm Spririt and Puck are not terrible heros but they are hard heroes to learn leading to low win rates. The differences in win rates arent too drastic with the highest chances being 58% however the difference between the top and bottom heros is 17%, meaning over a lot of games you will notice the difference.However just using the hero win rates as a predictor would still be bascally a coin toss.

We next made a similar plot that detailed which heroes were the most popular.

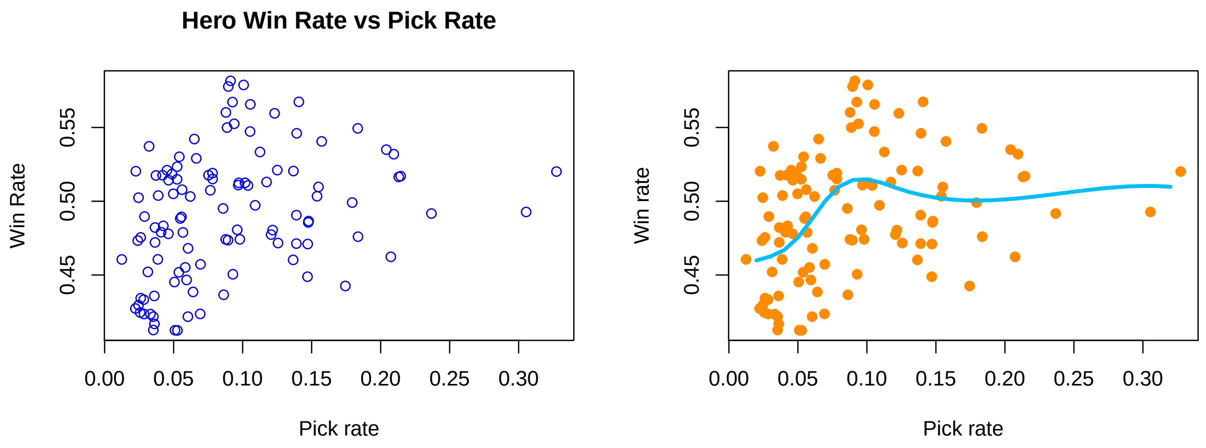
```
barplot(sort(acc, decreasing = TRUE), ylim = c(0, .35), xlab="Hero ID",ylab="pick Rate",col="blue",
main="Hero pick rate")
```



From this data, we see that Shadow Fiend, Windrunner, and Alchmist are the most picked with 32%, 30%, and 23%. This means that Shadow Fiend was in 32% of every game we looked, which is very high considering there are 100+ heros to chose from. The least picked were Jakiro, Lycan, and Abaddon each with about a 2% pick rate so a game with these heros in the line ups only has a 2% chance of happening.

```
par(mfrow = c(1,2))
plot(datat$acc, datat$accuracy, xlab="Pick rate",ylab="Win Rate",col="blue",
main="Hero Win Rate vs Pick Rate")

require(splines)
fit.bs = lm(accuracy ~ bs(acc, df=6), data=datat)
plot(datat$acc, datat$accuracy, pch = 19, xlab = "Pick rate", ylab = "Win rate", col = "darkorange")
lines(seq(0.02, .32,.01), predict(fit.bs, data.frame("acc"= seq(0.02, .32,.01))), col="deepskyblue", lt
```



For our next plot we hoped to capture the relationship between being picked and winning. The relationship is not that strong between the two but there is a slight trend. The lower picked heros do seem to have a slightly below average winrate. It seems to curve up and peak around a 10% pickrate with all the highest winrate heros lying there. Then it tappers of to around a 50% winrate the higher the pickrate. Overall, I would say the relationship between winrate and pick rate is weak

Proposed Analysis

For our analysis, we decided to focus on assigning a probability to "RadiantWin" taking a value of 1. Then, we would use this probability to assign values of 1 or 0 to the supplied testing data set, and then we would create a confusion matrix to see how well our models worked. For this paper, we focused on two different methods of this type of binary classification: Neural Network and Logistic Regression. Both of these methods will apply probabilities to our outcome variable, which we can then use to classify our results into each category.

Neural Network

To begin our analysis, we first wanted to try a nerual network. We passed our data through the a neural network, however we quickly ran into runtime issues. Therefore, we could only run a 3 layer network at the most as a 4-layer network would never complete or converge. We also could have a large hidden network as anything really more than 3 would take hours to run. We had to leave it at 3 hidden nodes because of that meaning the accuracy and internal representation were severely harmed. In the end using all the data as input we obtained a 66.01% accuracy. We believe if we had a more powerful computer and more time to run, we could add more layers and nodes to get a higher accuracy.

```r
library(neuralnet)
set.seed(42)
new_train_data = train_data[, -seq(8, 117)]
new_test_data = test_data[, -seq(8, 117)]
nn=neuralnet(`RadiantWin`~ ., data = train_data, hidden= 3,act.fct = "logistic",
                linear.output = FALSE, stepmax = 10000)

nn_conf_mat = table(test_data$RadiantWin, as.numeric(predict(nn,test_data)>=.5))

nn_conf_mat
```

```
##
##          0    1
##   0 3392 1417
##   1 1982 3209
```

```
(nn_conf_mat[1,1] + nn_conf_mat[2,2])/nrow(test_data)
```

```
## [1] 0.6601
```

Logistic Regression

Since, the neural network failed to accurately predict the winner. We moved onto logistic regression because logistic regression predicts a probability, so if the probability was greater a half we predicted a radiant win else a dire win. When we used all the variables we acquired an accuracy of 98.47% on the testing data, which is obviously really good. When we looked at the significances of the variables we found that the hero picks were mostly not significant and did not explain a lot. We currently have all the heroes as an indicator variable making the data really sparse maybe there could be a better way of representing it. With this information, we ran the logistic regression again but without the hero id data and with that we got 98.34% accuracy. Thus, removing the hero data did not impact the accuracy a lot. Some of the variables that were the most important predictors were the number of towers, number of barracks, deaths, team that got first blood, and kills by each team. Something interesting was the deaths were actually more significant than kills on first glance, we thought that they would be similar in significance. We believe that deaths are more important because they explain the deaths by external forces that are not enemy players such as creeps, towers, and jungle creeps. This is also the same reason why damage taken by a player is more significant than damage inflicted, because of non-player damage sources. The variables creep score and denies were not that significant compared to the others; because the roles (carry, support, and jungle) the players took was not considered in our data so the information carried by creep score and denies was diminished.

```
mylogit <- glm(`RadiantWin` ~ ., data = train_data, family = "binomial")

summary(mylogit)$coefficients[c(3:6,118:157),]
```

```
##                            Estimate  Std. Error     z value      Pr(>|z|)
## RadiantTowerStatus     1.0499470223 0.039170601  26.8044654 2.866281e-158
## DireTowerStatus       -1.0936200161 0.039285255 -27.8379256 1.508230e-170
## RadiantBarracksStatus  0.1892120348 0.039821513   4.7515029  2.019103e-06
## DireBarracksStatus    -0.0850005740 0.042083868  -2.0197900  4.340518e-02
## Player1Kills          -0.0670522655 0.032405919  -2.0691364  3.853329e-02
## Player2Kills          -0.0726816533 0.032408672  -2.2426606  2.491871e-02
## Player3Kills          -0.1036278134 0.032363142  -3.2020319  1.364619e-03
## Player4Kills          -0.0887235347 0.032770606  -2.7074121  6.781003e-03
## Player5Kills          -0.0728617354 0.032523906  -2.2402517  2.507459e-02
## Player6Kills           0.0945594854 0.032507880   2.9088174  3.627987e-03
## Player7Kills           0.1016033614 0.032664531   3.1105103  1.867644e-03
## Player8Kills           0.1012424698 0.033131048   3.0558185  2.244471e-03
## Player9Kills           0.0998158950 0.032572516   3.0644208  2.180919e-03
## Player10Kills          0.0926866522 0.032925588   2.8150341  4.877206e-03
## Player1Deaths         -0.2148844941 0.034618154  -6.2072777  5.391030e-10
## Player2Deaths         -0.1916411317 0.034776966  -5.5105765  3.576603e-08
## Player3Deaths         -0.1913037974 0.035094685  -5.4510761  5.006595e-08
## Player4Deaths         -0.2221305696 0.034666613  -6.4076225  1.478063e-10
## Player5Deaths         -0.1965125301 0.034497217  -5.6964749  1.223099e-08
## Player6Deaths          0.1927346450 0.034062955   5.6581893  1.529783e-08
## Player7Deaths          0.2048603192 0.034835556   5.8807822  4.083323e-09
## Player8Deaths          0.1760091058 0.034114581   5.1593512  2.478070e-07
## Player9Deaths          0.1763345526 0.034440636   5.1199564  3.056063e-07
## Player10Deaths         0.1853989885 0.034530016   5.3692123  7.908128e-08
## Player1CS             -0.0002479004 0.001342026  -0.1847209  8.534479e-01
## Player2CS              0.0022197001 0.001385223   1.6024132  1.090643e-01
## Player3CS             -0.0005516659 0.001445010  -0.3817729  7.026298e-01
## Player4CS              0.0015609994 0.001198547   1.3024100  1.927763e-01
## Player5CS              0.0018383431 0.001384774   1.3275401  1.843301e-01
## Player6CS             -0.0011582257 0.001345997  -0.8604966  3.895154e-01
## Player7CS             -0.0008338130 0.001322632  -0.6304193  5.284203e-01
## Player8CS             -0.0002989676 0.001374555  -0.2175015  8.278176e-01
## Player9CS              0.0009413542 0.001426216   0.6600361  5.092306e-01
## Player10CS            -0.0009628568 0.001469025  -0.6554393  5.121849e-01
## Player1Denies         -0.0056678418 0.008239262  -0.6879065  4.915117e-01
## Player2Denies         -0.0144210846 0.008443790  -1.7078925  8.765629e-02
## Player3Denies         -0.0111728250 0.008406154  -1.3291245  1.838069e-01
## Player4Denies         -0.0143120398 0.008449884  -1.6937558  9.031170e-02
## Player5Denies         -0.0029619565 0.008536793  -0.3469636  7.286187e-01
## Player6Denies          0.0013637935 0.008346086   0.1634052  8.701994e-01
## Player7Denies         -0.0071754372 0.008388902  -0.8553488  3.923581e-01
## Player8Denies         -0.0128129610 0.008810010  -1.4543640  1.458454e-01
## Player9Denies         -0.0104330515 0.008967443  -1.1634366  2.446524e-01
## Player10Denies        -0.0106595297 0.008465264  -1.2592082  2.079551e-01
```

```r
log_conf_mat=table(test_data$RadiantWin, as.numeric(predict(mylogit,test_data)>=.5))

log_conf_mat
```

```
##
##         0    1
##    0 4752   57
##    1   96 5095
```

```r
(log_conf_mat[1,1] + log_conf_mat[2,2])/nrow(test_data)
```

```
## [1] 0.9847
```

```r
mylogitnh <- glm(`RadiantWin` ~ ., data = new_train_data, family = "binomial")

log_conf_mat_nh=table(new_test_data$RadiantWin, as.numeric(predict(mylogitnh,new_test_data)>=.5))

log_conf_mat_nh
```

```
##
##         0    1
##    0 4748   61
##    1  105 5086
```

```r
(log_conf_mat_nh[1,1] + log_conf_mat_nh[2,2])/nrow(new_test_data)
```

```
## [1] 0.9834
```

Regression output for the data without heroes is omitted due to being approximately the same, since the hero data didn't add much to our prediction ability.

Conclusion and Discussion

Conclusion

From our use of a Logistic Regression, we found that the best indicators for which team won a game were deaths, wins, and the structure status status for each team. We found that variables such as times chatted and the creep score and denies were rather insignificant when predicting the outcome. Our logistic regression was incredibly good at classifying who won the match, with a less than 2% error rate for both of our models. The neural net performed a lot worse, having a 38.35% error rate.

Discussion

Our project is not without issues. One thing our data set does not identify is the role of each player. In DOTA, each player generally plays a different role in the game for their team. However, our original JSON file did not contain the information pertaining to the role the player played, as in DOTA this is not set by the game but by the players themselves. This may be the reason why creep score and denies were insignificant, among other things. Perhaps if we created an accumulation variable per team we would see a more significant difference. Additionally, our "times chatted" variable did not differentiate between who sent the message, but rather just counted the number of messages sent in the game. As a result, it doesn't entirely capture the effect chatting has on a team's ability to win. It could also be argued that "times chatted" depends on which side is winning, rather than how we present it in the model. This opens up the door to future analysis on how chatting affects (or is affected by) how a team is performing in game.

Additionally, we wanted our neural net to have more hidden nodes, but we could not support the computational time needed to do so with our data. If we cut down the size of our data set a bit we may have been able to get a neural net that provided a better fit than the one we used.

If we had the computational ability and time to do so, we would also be interested in looking at if certain combinations of heroes were significant to determining whether or not a team won. There are instances where certain heroes synergize well when played together. There is also the case where certain heroes counter each other in game, so that could change the winrate as well. Thus, it would be extremely interesting to see if these hero combinations changed our analysis at all, especially the logistic regression with hero-level data.

References

Kinkade, Nicholas and Kyung yul Kevin Lim (n.d): “DOTA 2 Win Prediction” *UC San Diego* 1-7.

“OpenDota Data Dump”, OpenDota Blog. Retrieved from
https://blog.opendota.com/2015/12/20/datadump/