Python Project

League of Legends Data Analysis

Course: Scientific Programming with Python

Lecturer: Eyal Nussbaum

Student: Keren Halpert (313604621)

GitHub Repository: https://github.com/halpert-keren/Python_LOL_Analysis
Presentation Video: https://drive.google.com/file/d/1PCyjMqHgpon8gunI1TWXUIHnPJG-TzET/view?usp=sharing

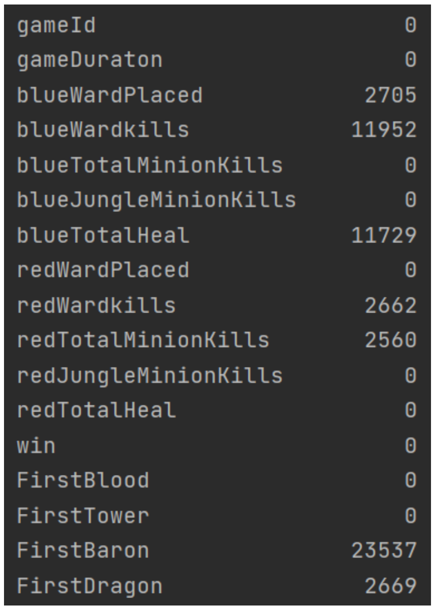
1. INTRO

League of Legends (abbreviated LoL) is a multiplayer online battle arena video game developed and published by Riot Games.
In the game, two teams of five players battle in player versus player combat, with each team occupying and defending their own half of the map.
Each of the ten players controls a character, known as a "champion", with unique abilities and differing styles of play.
During a match, champions collect experience points to gain levels and purchase items in order to defeat the opposing team.
In the game's main mode, Summoner's Rift, a team wins by pushing through to the enemy base and destroying their "nexus", a large structure located within it.
During the game, the players also fight AI controlled minions, place vision wards to deal with fog of war, and support each other with healing and shielding abilities.

The dataset provided includes 17 columns, and 65,896 rows.
Of the 17 columns, one is a game id value, five are categorical and the remaining 11 are continuous numerical.
The categorical columns have the same 2 value options: Red and Blue. These colors represent the 2 teams participating in the game.
One of the five represents the winning team and the target class for this dataset.
The other 4 represent the team that was first in achieving the topic (first team to slay a dragon, first team to kill an opposing team member...).
The eleven continuous numerical columns can be divided as follows:

> One of the 11 represents the duration of the particular game in seconds.
> The remaining 10 represent team achievements like the number of wards placed by the team, number of minions killed and so on and so forth.
> Of the ten, there are five for each of the two teams.

2. INITIAL DATA ANALYSIS



```
gameId                     0
gameDuraton                0
blueWardPlaced          2705
blueWardkills          11952
blueTotalMinionKills       0
blueJungleMinionKills      0
blueTotalHeal          11729
redWardPlaced              0
redWardkills            2662
redTotalMinionKills     2560
redJungleMinionKills       0
redTotalHeal               0
win                        0
FirstBlood                 0
FirstTower                 0
FirstBaron             23537
FirstDragon             2669
```

The dataset is missing values only in some of the columns as shown in figure 2.1, which depicts the number of values missing from every column.
The missing values shall be filled with either the mean value (in the event of a continual numerical column) or the mode (in the event of a categorical column).
The gameId column has been erased from the dataset since it is not relevant for the analysis.

When dealing with different types of data, it is best to divide the methods.

Thus, I shall start with the categorical columns.
As mentioned above in section 1. INTRO, there are five categorical columns, of which only two have missing data as shown in figure 2.1.
The missing values have been replaced by the mode of the column (in both cases the mode is “Red”). However, since there are so many missing value, we must check in what way this is impacting the data.
We shall start by looking at the distribution of the value in correlation with the target class (“win”) to see if there are inconsistencies after replacing the values.

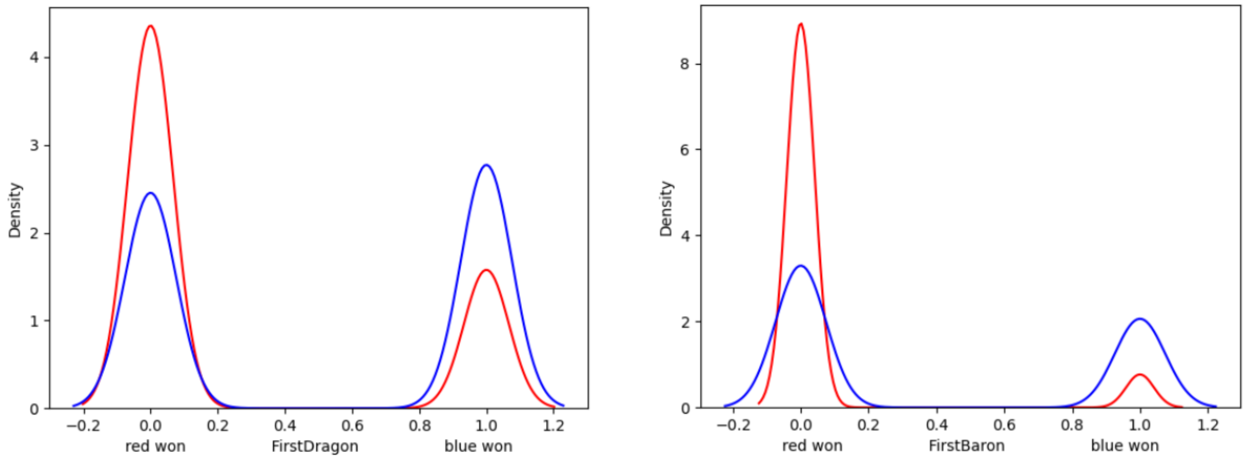


figure 2.2

The plots in figure 2.2 show the distribution of the values in both columns. We can tell that when the blue team won the blue team was able to both slay the first dragon and also kill the first baron, and the same goes for the red team. However, it is obvious that when the red team won, the number of slain dragons and the number of killed barons is significantly higher.

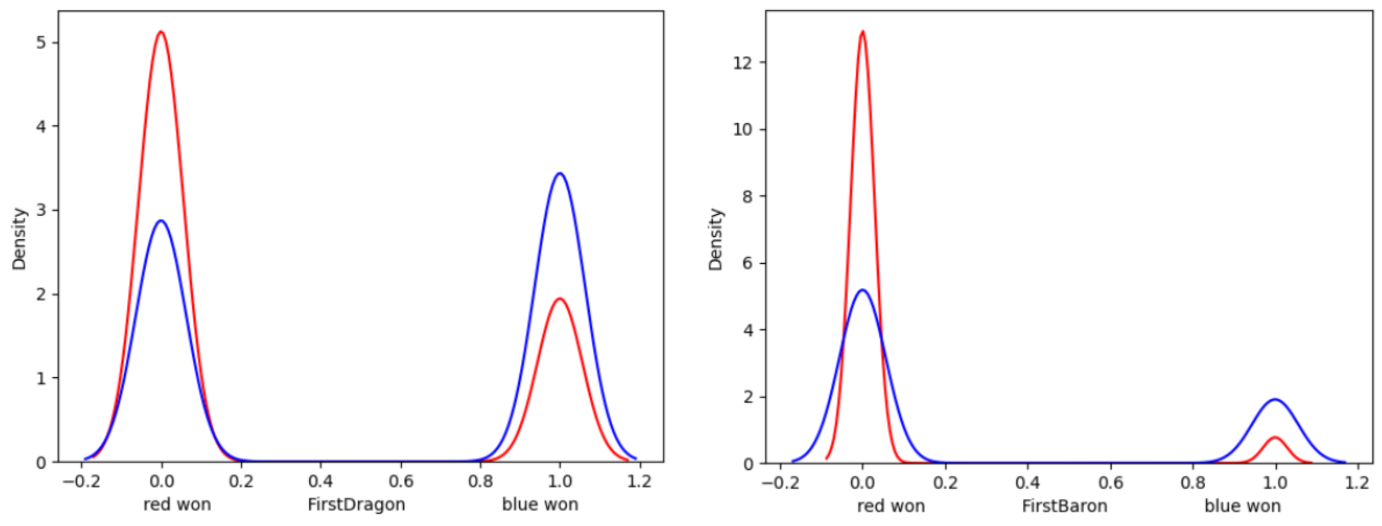
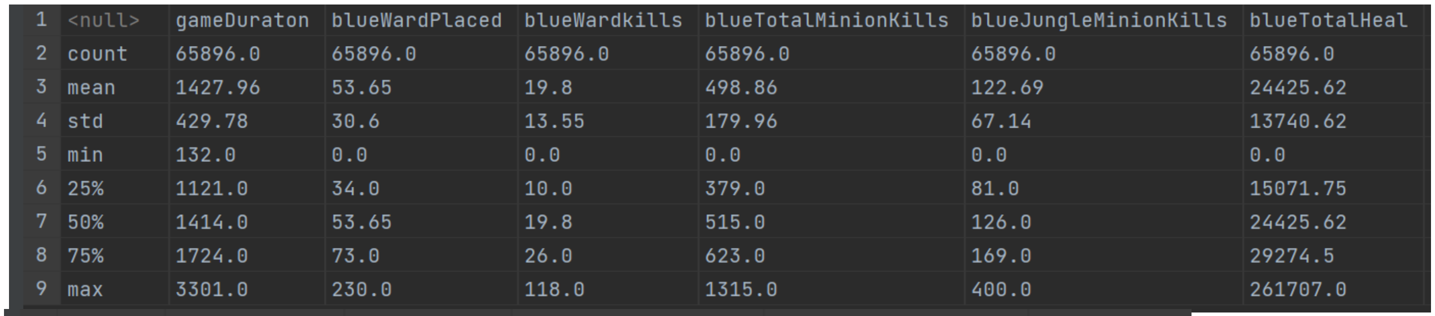


figure 2.3

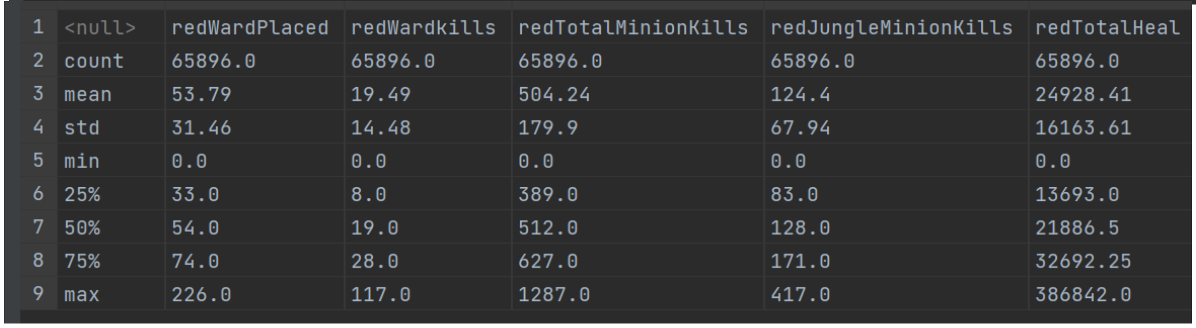
These plots in figure 2.3, show the same distribution, AFTER the replacement of the missing values. The plots are so similar that it is evident that the change (even considering that there are 34% missing value in the “FirstBaron” column) is inconsequential. Therefor I see no problem with the replacements.

After dealing with the categorical value, we shall move on to the numerical.

The information provided by the pandas describe method (shown in figure 2.4) can be used to identify any outliers in the data. However, in this particular dataset, it seems as though every one of the columns contains outliers since the “max” value is significantly higher than the rest of the values in every one of the columns, and the “min” value is significantly lower than the rest also.



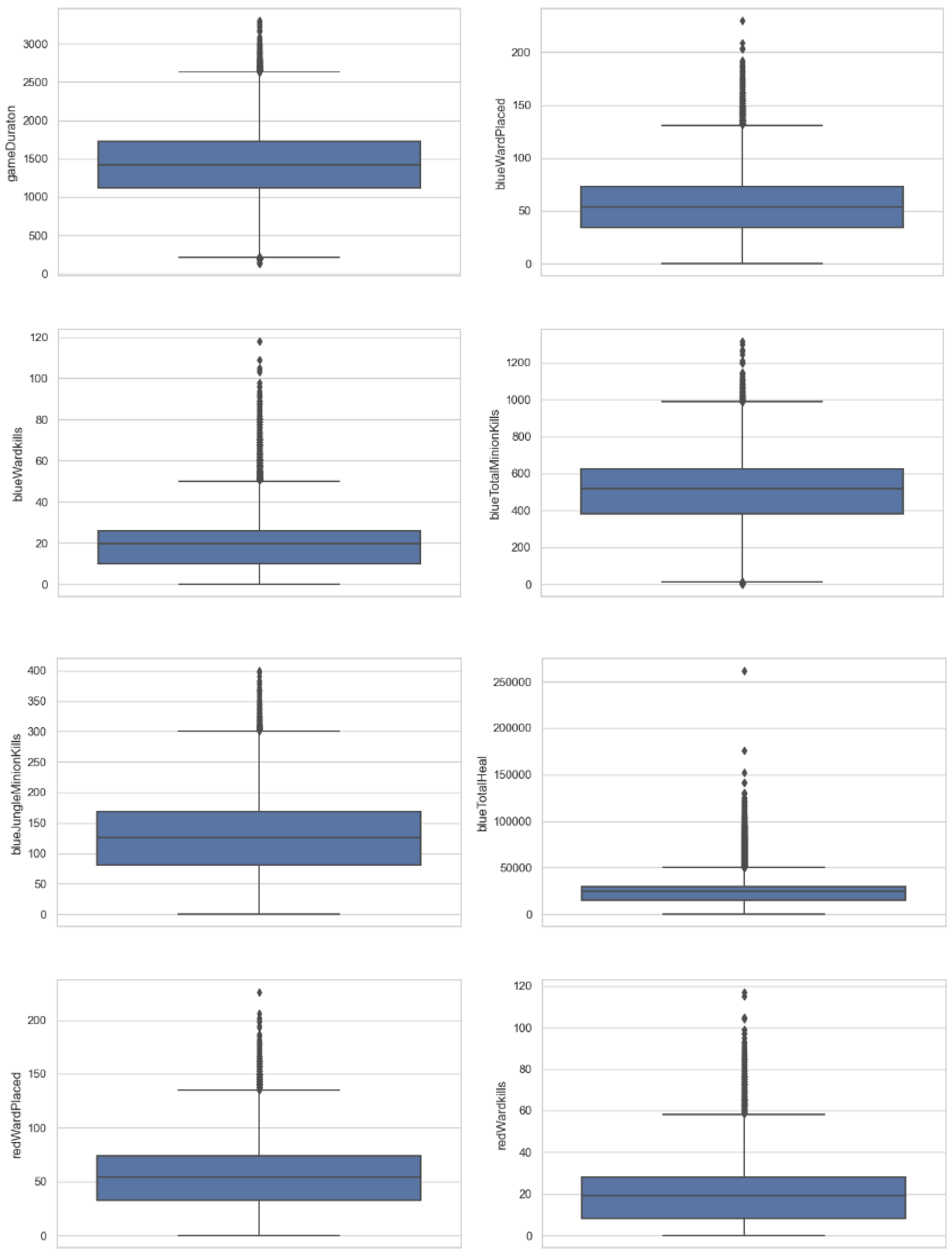
<null>	gameDuraton	blueWardPlaced	blueWardkills	blueTotalMinionKills	blueJungleMinionKills	blueTotalHeal
count	65896.0	65896.0	65896.0	65896.0	65896.0	65896.0
mean	1427.96	53.65	19.8	498.86	122.69	24425.62
std	429.78	30.6	13.55	179.96	67.14	13740.62
min	132.0	0.0	0.0	0.0	0.0	0.0
25%	1121.0	34.0	10.0	379.0	81.0	15071.75
50%	1414.0	53.65	19.8	515.0	126.0	24425.62
75%	1724.0	73.0	26.0	623.0	169.0	29274.5
max	3301.0	230.0	118.0	1315.0	400.0	261707.0



<null>	redWardPlaced	redWardkills	redTotalMinionKills	redJungleMinionKills	redTotalHeal
count	65896.0	65896.0	65896.0	65896.0	65896.0
mean	53.79	19.49	504.24	124.4	24928.41
std	31.46	14.48	179.9	67.94	16163.61
min	0.0	0.0	0.0	0.0	0.0
25%	33.0	8.0	389.0	83.0	13693.0
50%	54.0	19.0	512.0	128.0	21886.5
75%	74.0	28.0	627.0	171.0	32692.25
max	226.0	117.0	1287.0	417.0	386842.0

figure 2.4

For this reason, it is important to investigate the outliers further. This shall be done with the use of boxplots (as shown in figure 2.5).


gameDuraton
0
500
1000
1500
2000
2500
3000
blueWardPlaced
0
50
100
150
200
blueWardkills
0
20
40
60
80
100
120
blueTotalMinionKills
0
200
400
600
800
1000
1200
blueJungleMinionKills
0
50
100
150
200
250
300
350
400
blueTotalHeal
0
50000
100000
150000
200000
250000
redWardPlaced
0
50
100
150
200
redWardkills
0
20
40
60
80
100
120
figure 2.5

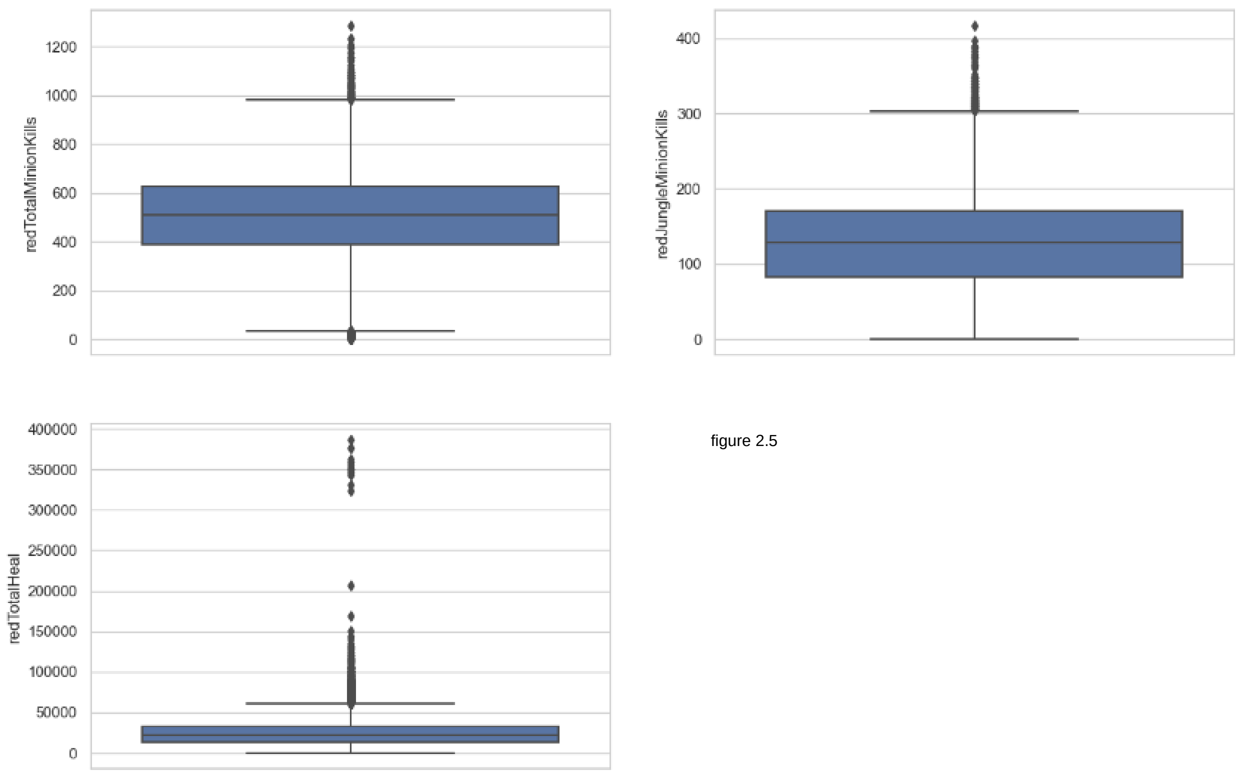


figure 2.5

As suspected, the boxplots show many outliers in every one of the columns. To further investigate the amount of zeros in the data specifically, the histogram/KDE plot comes in handy. Below we can tell that 6 of the columns ((red & blue)WardPlaced, (red & blue)JungleMinionKills, (red & blue)WardKills) have an exorbitant amount of zero values that seem very inconsistent with the bell shape of the rest of the value distribution.

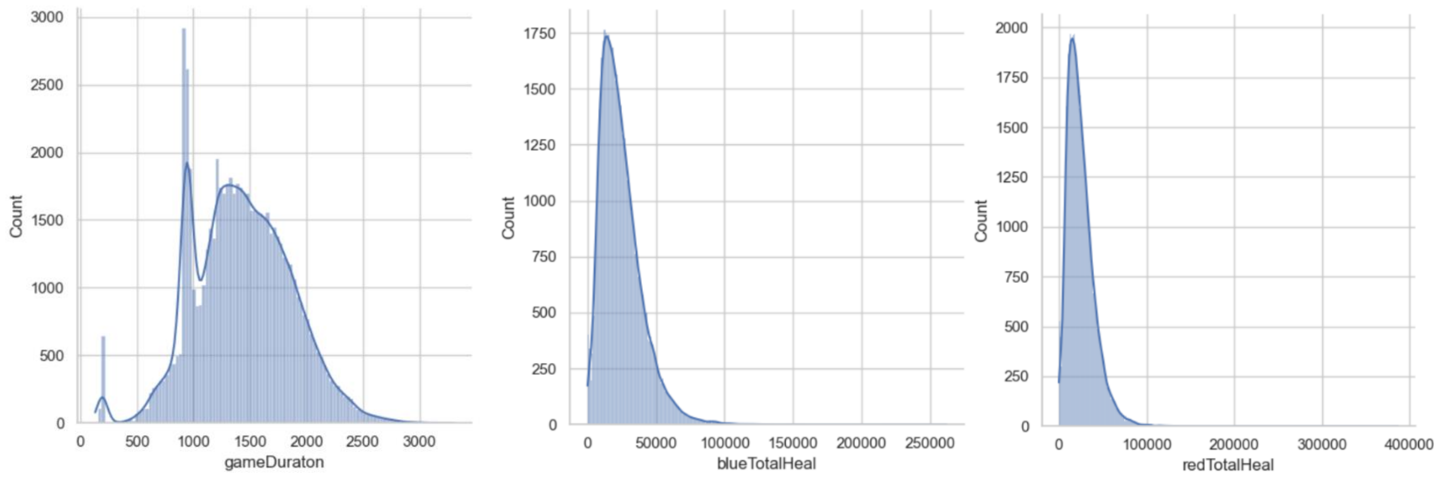


figure 2.6

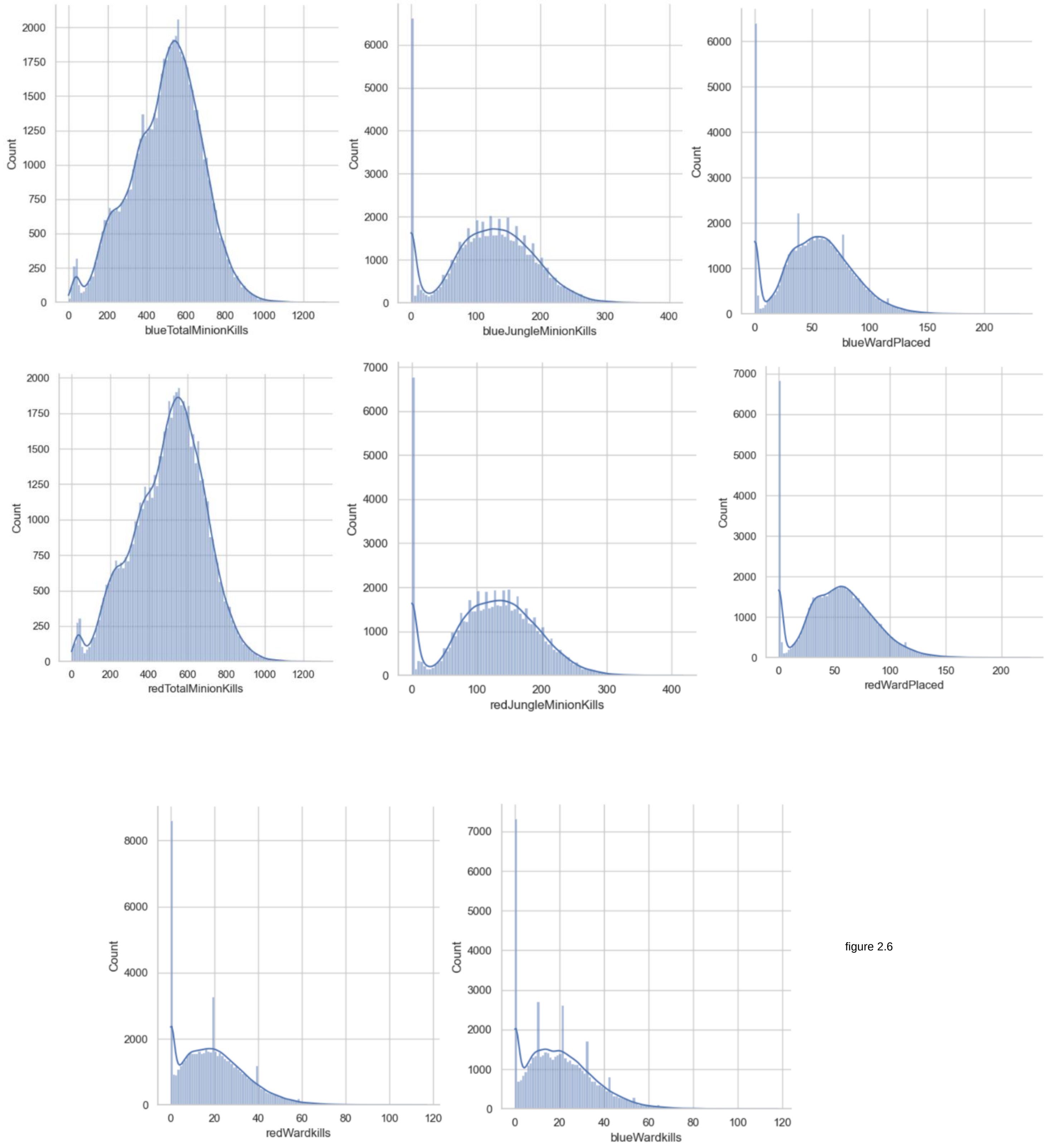

Count
blueTotalMinionKills
blueJungleMinionKills
blueWardPlaced
redTotalMinionKills
redJungleMinionKills
redWardPlaced
redWardkills
blueWardkills

figure 2.6

Looking at these plots, the question that arises is whether the zeros in the 6 columns are in the same rows.

The answer is that there are exactly 4,671 rows that contain zeros in all six of the aforementioned columns. This translates to roughly 7% of the whole dataset. While it might make some sense that all 6 columns contain zeros if the game ended extremely quickly (because the players might not have needed to use wards or kill jungle minions – the topics of the 6 columns), this is not the case. The next plot (figure 2.7) shown the distribution of the game duration of the 4,671 games in which there are zero wards placed, zero wards killed and zero jungle minions killed.

We can tell by this that the game duration distribution is similar to that of the distribution according to the whole dataset and this does not make sense when considering that there were no wards used or jungle minions killed at all.

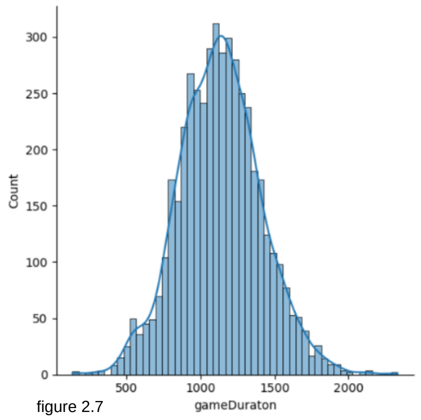


figure 2.7

For this reason, I have decided that these rows that contain 6 zeros in the columns discussed shall be removed completely from the dataset, as they appear to be errors.

Now, we shall check again for blatant outliers using box plots.

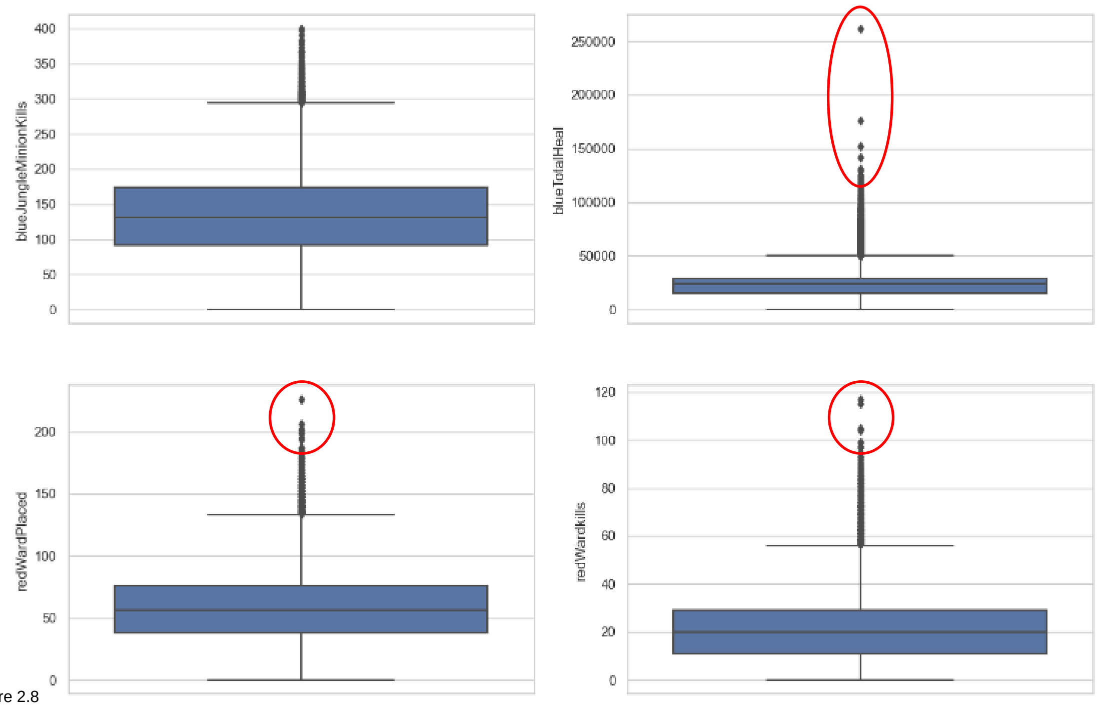


figure 2.8

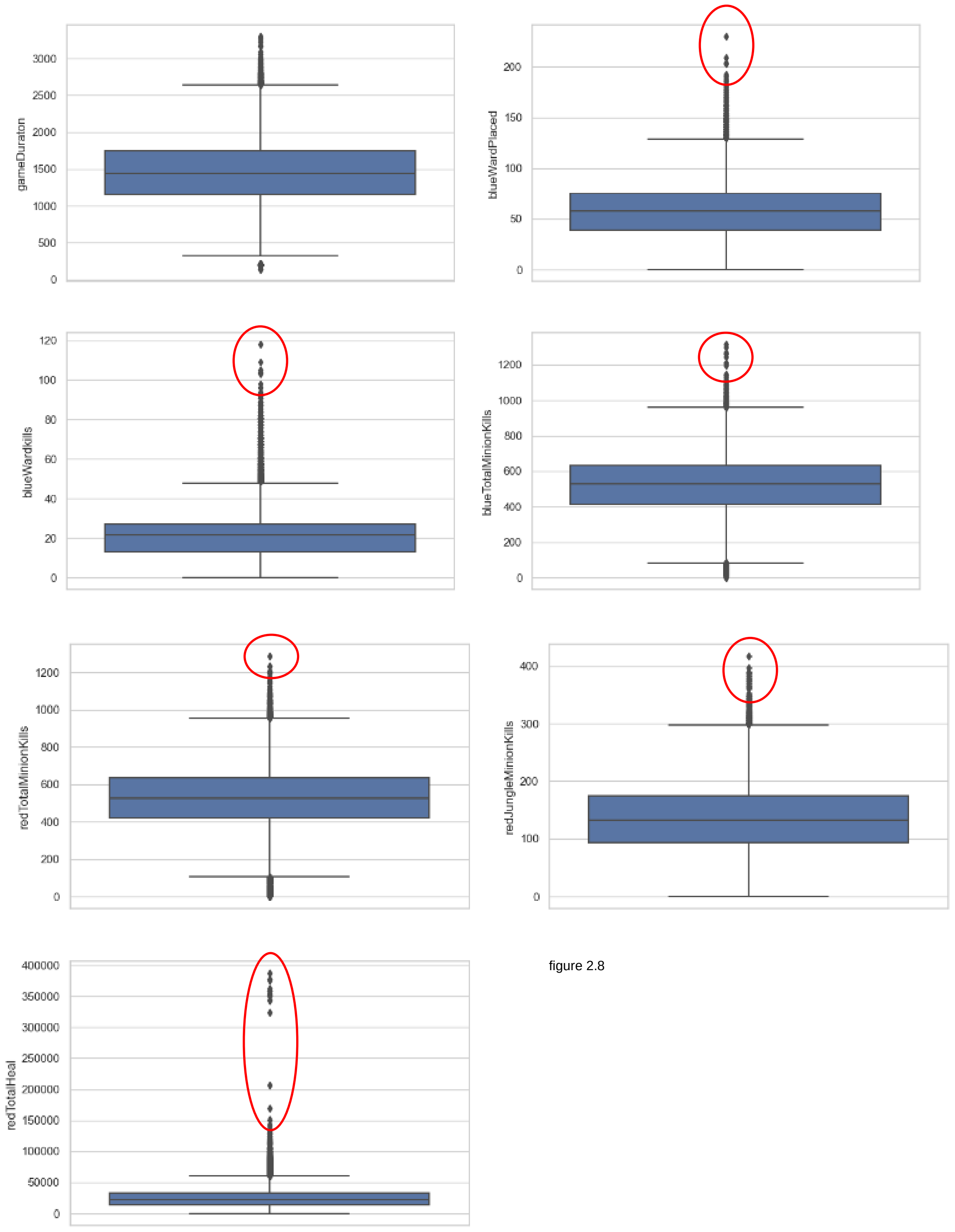


figure 2.8

The red circles represent the cutoff section, and these rows shall be considered blatant outliers and therefor removed from the dataset.

As of now, the dataset contains 16 columns and 61,161 rows (over 92% of the original dataset).

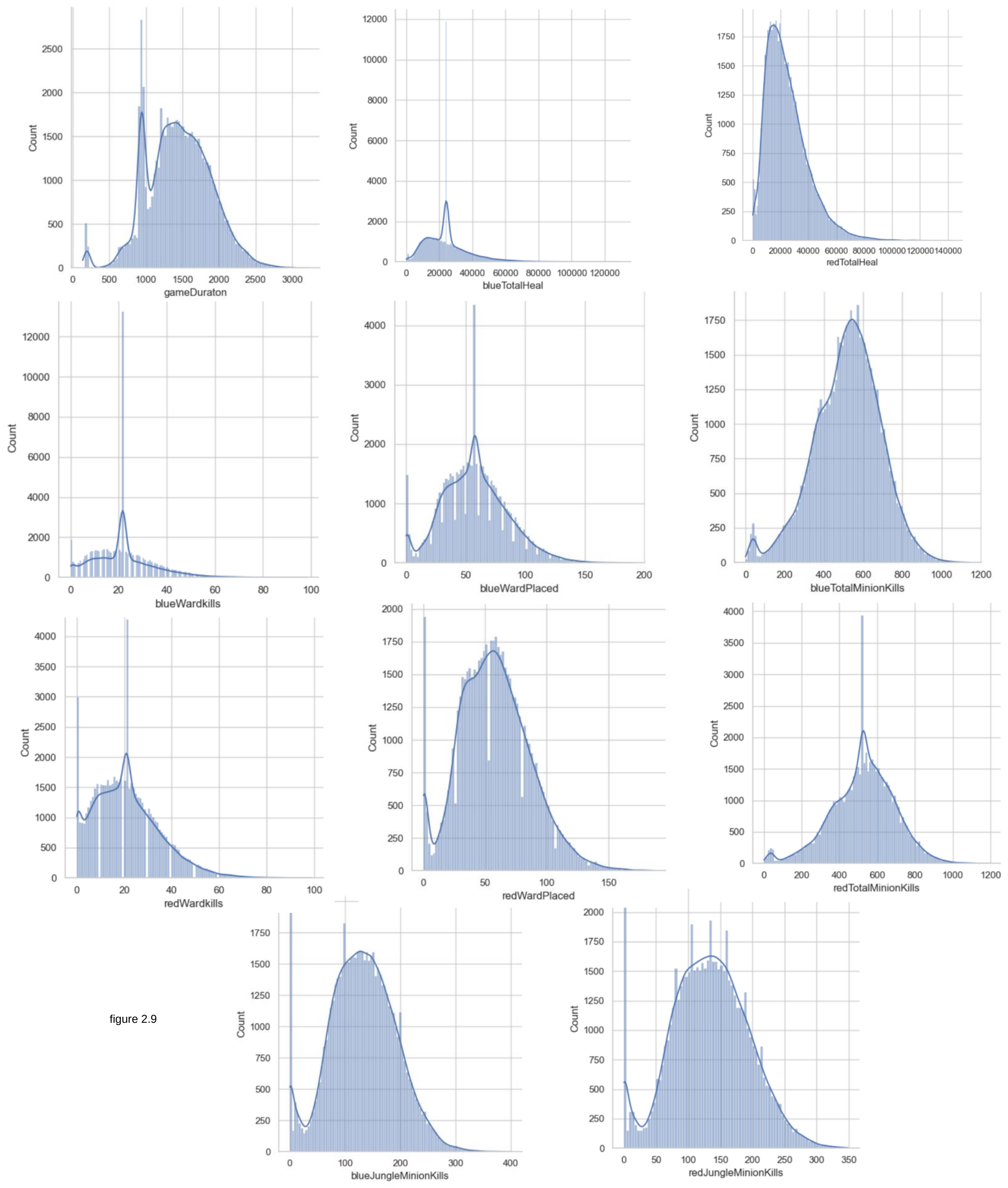


figure 2.9

It is fairly easy to tell that the general distribution of the data is very similar in every pair of features (red and blue of the corresponding columns). The major differences are the average points in the five columns that according to figure 2.1 had the most amount of missing data and therefor were filled with the column average.
At this point, all changes are saved in the new CSV file: “clean_lol.csv”. Additionally, typos in column names have been fixed ('gameDuraton' to 'gameDuration', 'blueWardkills' to 'blueWardKills', 'redWardkills' to 'redWardKills').

3. EXPLORATORY DATA ANALYSIS

Feature correlation shall be the first step in trying to analyze the now clean data.

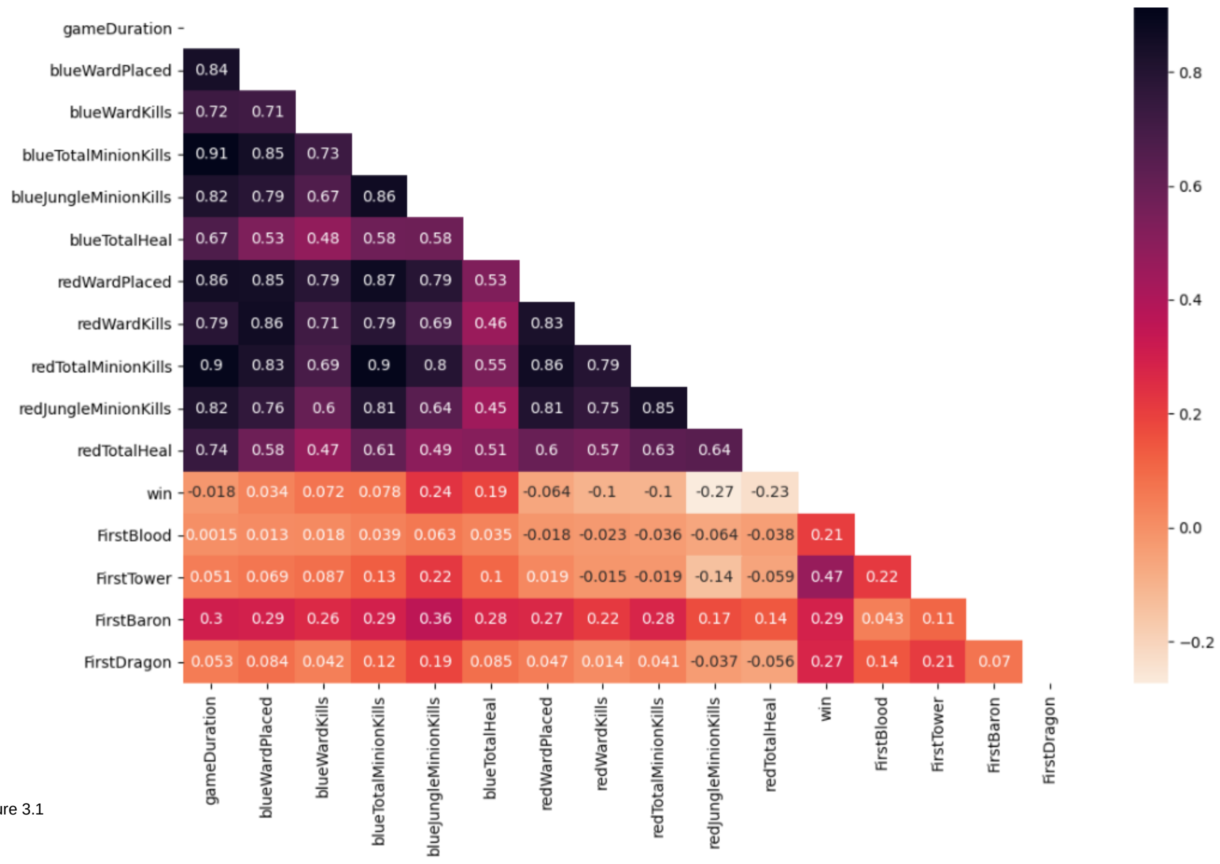


figure 3.1

We learn from the heatmap in figure 3.1 that there are strong correlations between many of the numerical features, however it seems that there is little correlation with our target class, “win”. In order to understand the correlations further, we shall look to each of the features with regards to the target.

First, gameDuration, as shown in figure 3.2, depicts that there is little to no correlation between the length of the game and the winning team. We can tell that the number of games won by any given team has the same probability regardless of the game duration.

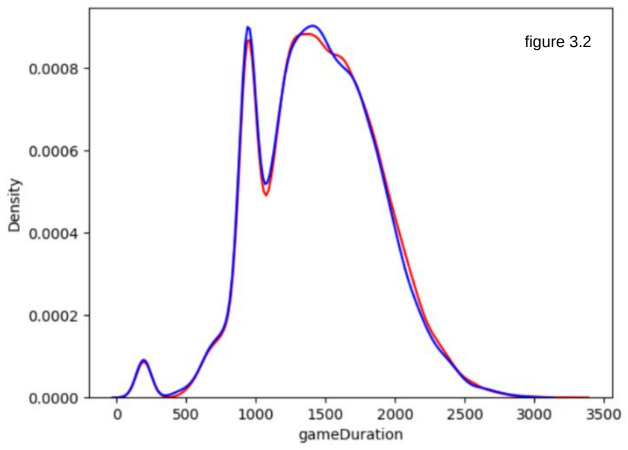


figure 3.2

Figure 3.3 shows scatter plots between the five numerical features regarding the Red team. Here we notice that the “redJungleMinionKills” feature has a unique distribution between the red and blue dots, which depict the winning team of the specific game. This division shows that the games where the red team won, they were the team that was able to kill more jungle minions that the opposing blue team.

Unfortunately, the rest of the features are not as informative at first glance.

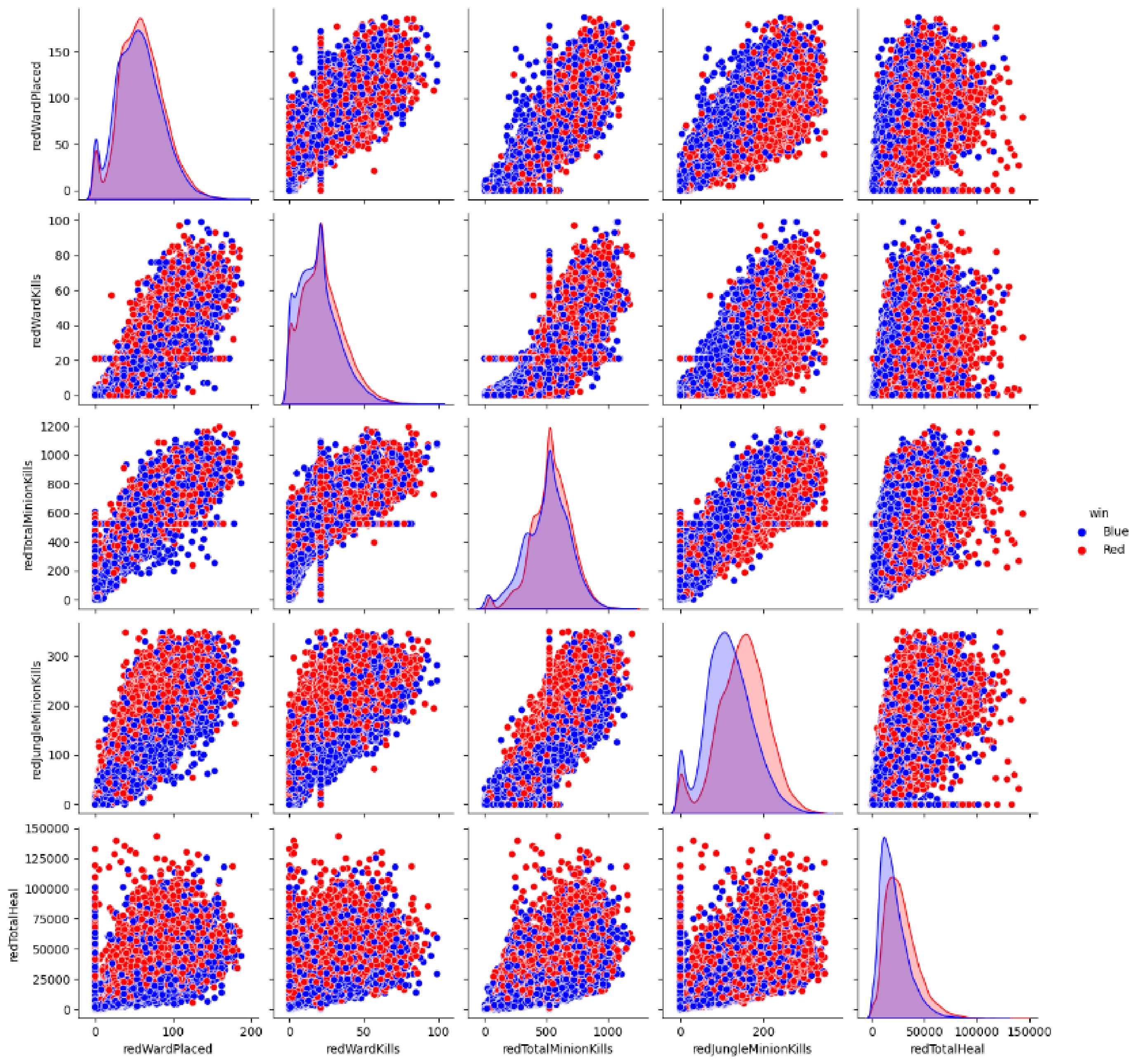


figure 3.3

As with the Red team, also the blue team shows similar tendencies, both with the “blueJungleMinionKills” and with the rest of the features that are not as divided.

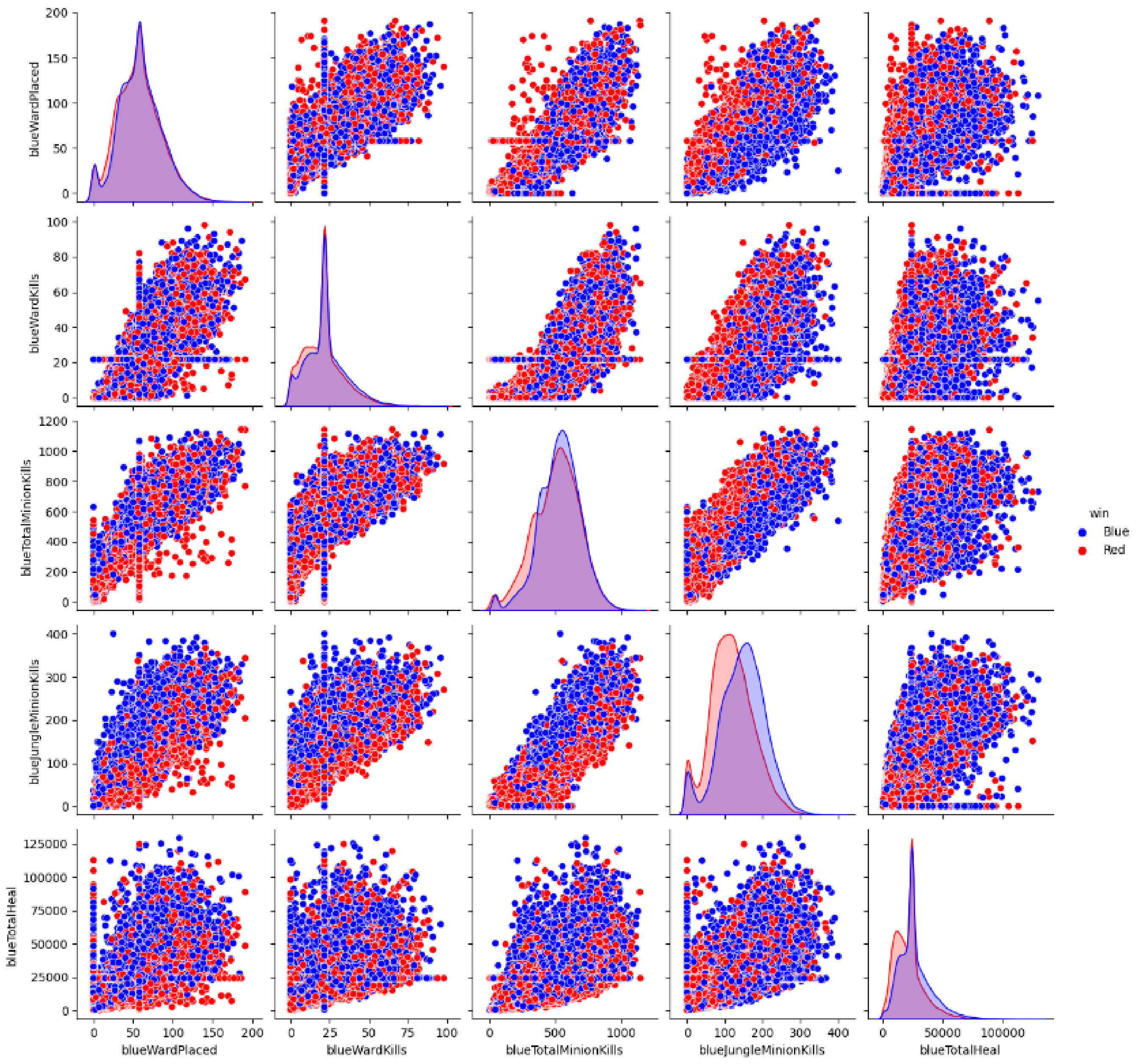


figure 3.4

The next set of plot (figure 3.5 and 3.6) shows the distribution by pair in accordance with the different team.

In both, we can tell that, again, the “jungleMinionKills” seems to be a rather good feature in terms of separation between the Blue and Red team.

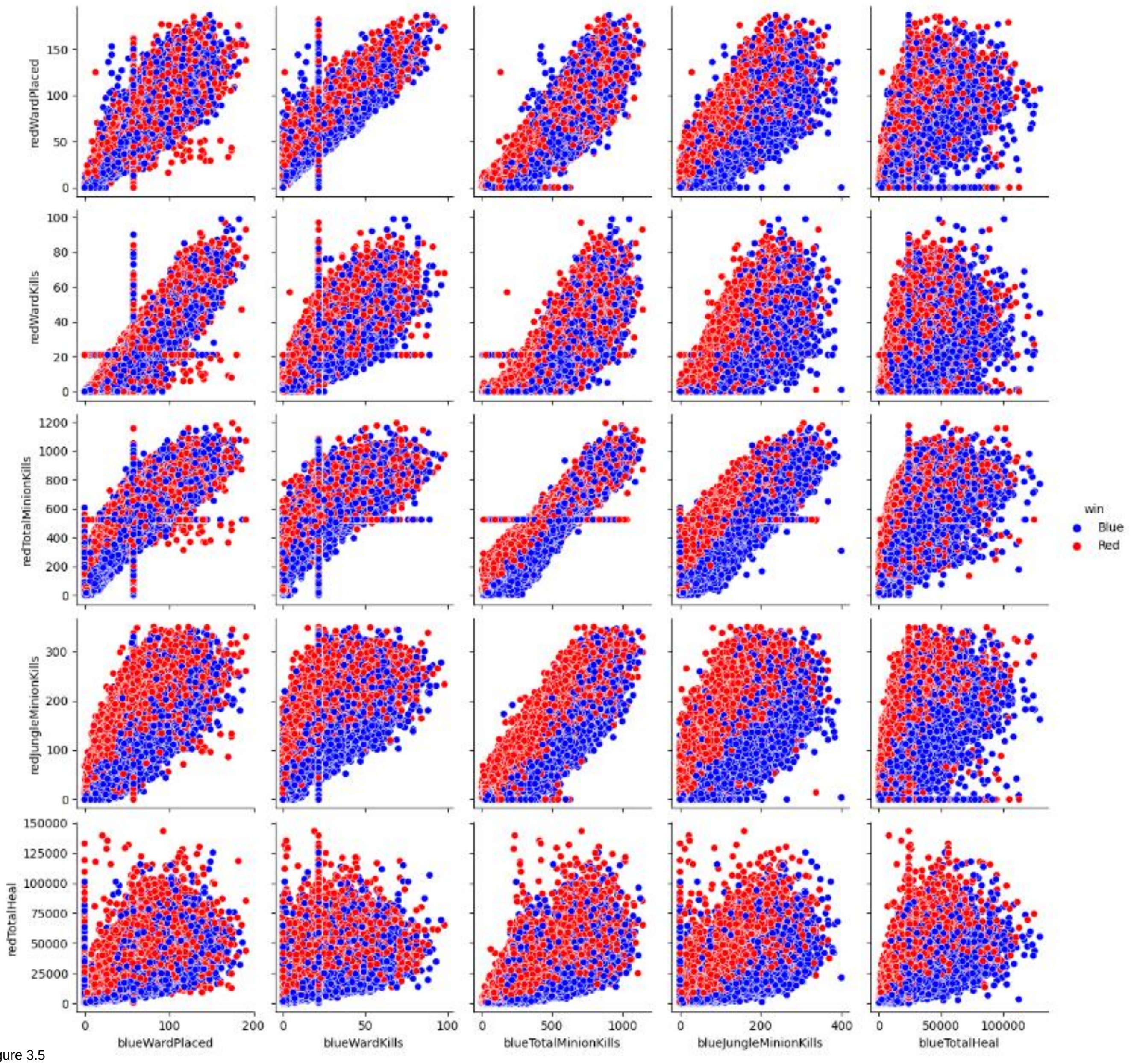


figure 3.5

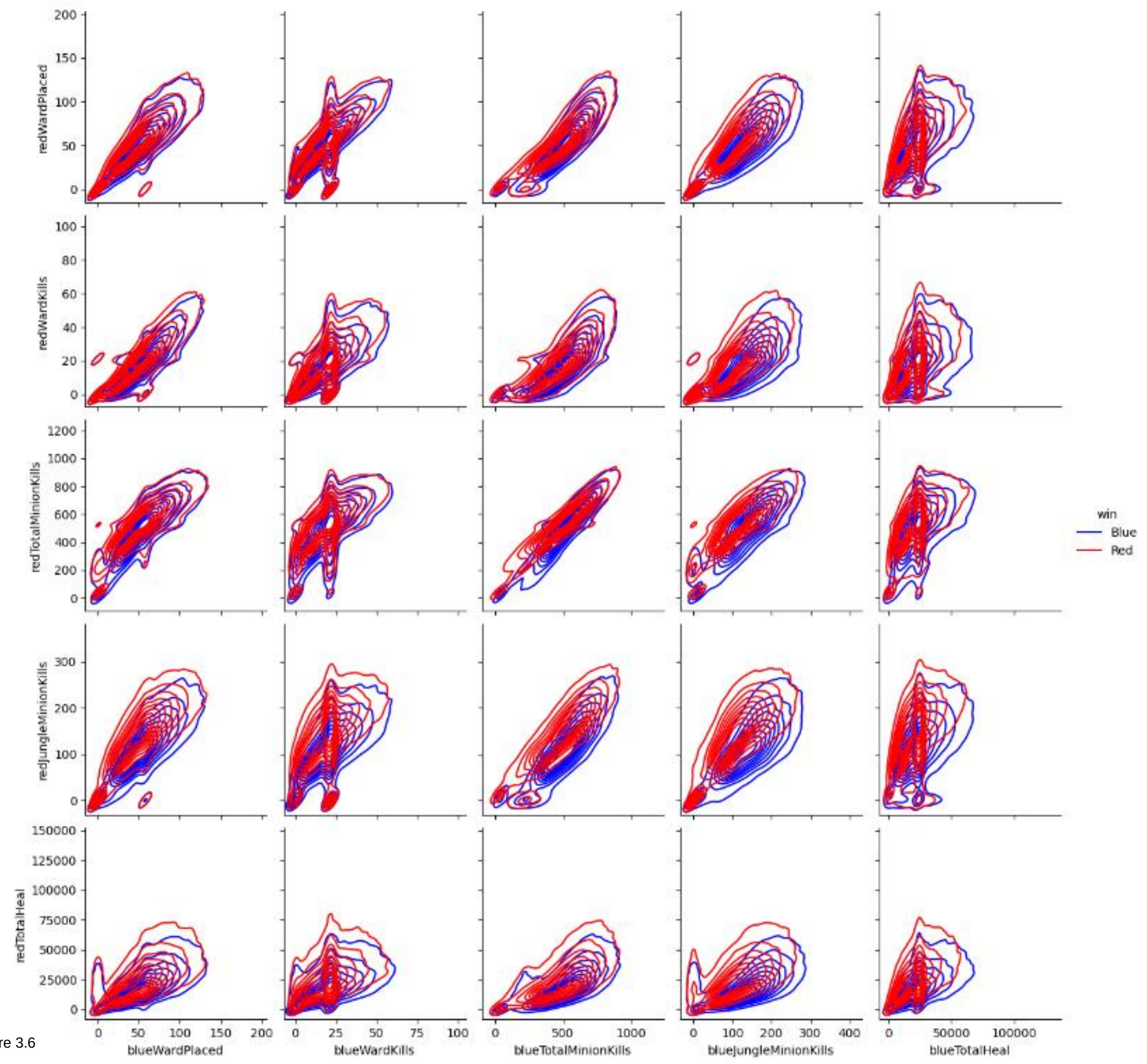

redWardPlaced
redWardKills
redTotalMinionKills
redJungleMinionKills
redTotalHeal
blueWardPlaced
blueWardKills
blueTotalMinionKills
blueJungleMinionKills
blueTotalHeal
win
Blue
Red

figure 3.6

In order to represent the data in different ways, I will focus on the fact of the 11 numerical columns, there are 10 that are pairs of the data, representing the 2 opposing teams. Because of this, it might be fruitful to change the division to winner vs. loser rather than Red vs Blue. Therefore, the new columns will be:

1. winnerWardPlaced (comprised of ‘blueWardPlaced’, ‘redWardPlaced’)
2. loserWardPlaced (comprised of ‘blueWardPlaced’, ‘redWardPlaced’)
3. winnerWardKills (comprised of ‘blueWardKills’, ‘redWardKills’)
4. loserWardKills (comprised of ‘blueWardKills’, ‘redWardKills’)
5. winnerTotalMinionKills (comprised of ‘blueTotalMinionKills’, ‘redTotalMinionKills’)
6. loserTotalMinionKills (comprised of ‘blueTotalMinionKills’, ‘redTotalMinionKills’)
7. winnerJungleMinionKills (comprised of ‘blueJungleMinionKills’, ‘redJungleMinionKills’)
8. loserJungleMinionKills (comprised of ‘blueJungleMinionKills’, ‘redJungleMinionKills’)
9. winnerTotalHeal (comprised of ‘blueTotalHeal’, ‘redTotalHeal’)
10. loserTotalHeal (comprised of ‘blueTotalHeal’, ‘redTotalHeal’)

Each of the new columns shall contain the value of either the Red or the Blue columns that comprise the new column, according to the winning (or losing) team. (i.e. If the red column contains the value 3, the blue column contains the value 7 and the ‘win’ column contains the value “Red”, then the value that shall be in the winner column is: 3, and the loser column: 7).

Using boxplots it is easy to compare the distribution changes between the winner and loser.

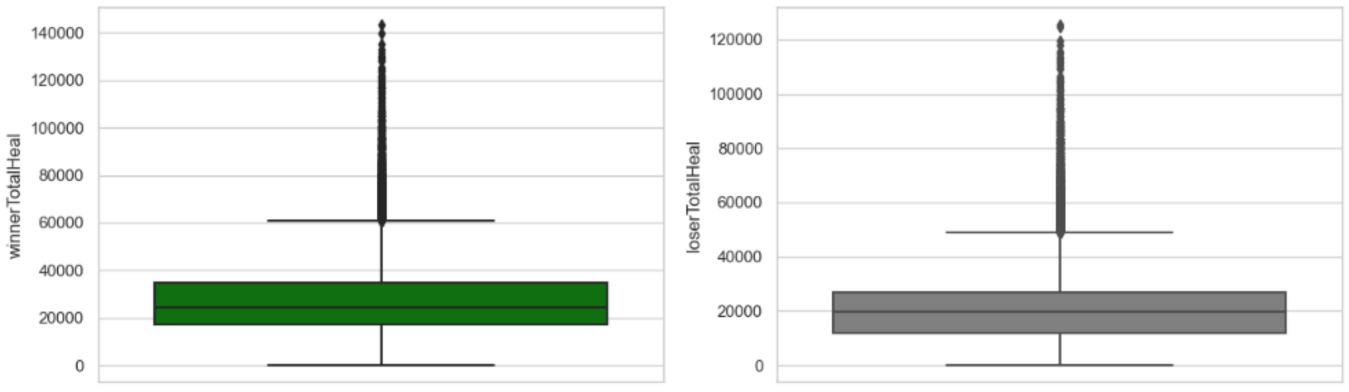


figure 3.7

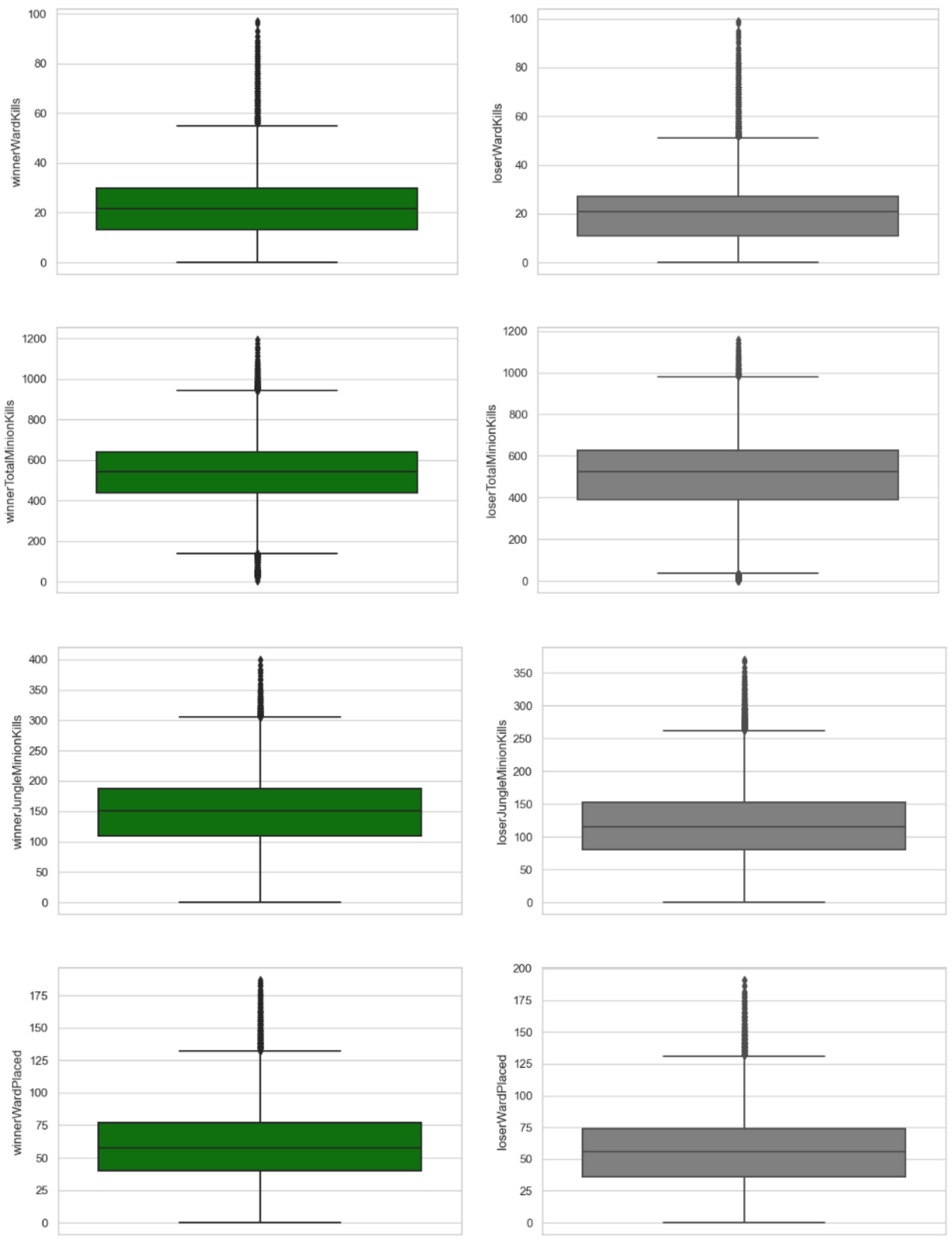
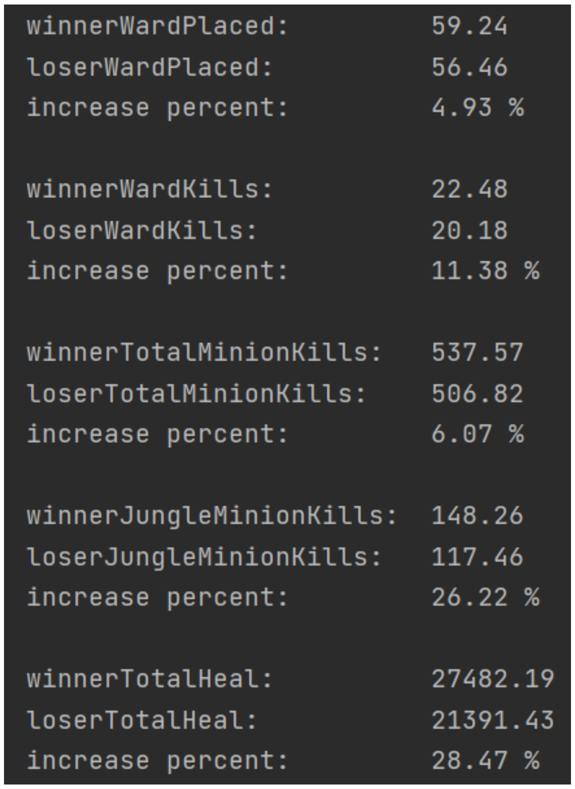

winnerWardKills
loserWardKills
winnerTotalMinionKills
loserTotalMinionKills
winnerJungleMinionKills
loserJungleMinionKills
winnerWardPlaced
loserWardPlaced

figure 3.7

While there are differences between the winner and the loser distributions in figure 3.7, it is rather hard to tell the exact variance.
Therefore we shall look at the statistical evaluations presented in figure 3.8, which displays the average values of each of the new columns as well as the percentage of increase between the two.
It is easy to see that the two features that seem to be the most impacted by the losing team and the winning are the “totalHeal” and the “jungleMinionKills” features that have an increase of over 26 percent and 28 percent respectively.



```
winnerWardPlaced:          59.24
loserWardPlaced:           56.46
increase percent:          4.93 %

winnerWardKills:           22.48
loserWardKills:            20.18
increase percent:          11.38 %

winnerTotalMinionKills:    537.57
loserTotalMinionKills:     506.82
increase percent:          6.07 %

winnerJungleMinionKills:   148.26
loserJungleMinionKills:    117.46
increase percent:          26.22 %

winnerTotalHeal:           27482.19
loserTotalHeal:            21391.43
increase percent:          28.47 %
```

figure 3.8

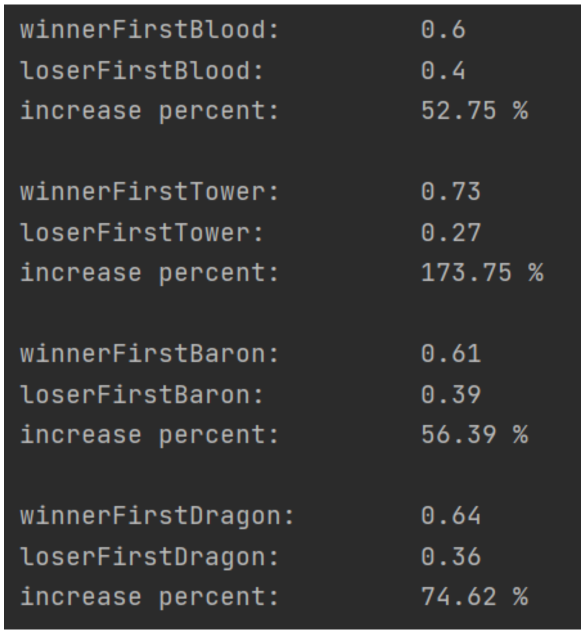
From the above findings we can determine that of the numerical features, it seems that the two features of jungle kills and health are the most important to the target question of who shall be the winning team. Also. To a lesser degree the ward kills feature.

At this point we shall take a closer look at the categorical columns.

In a similar fashion as has been done with the numerical features, the categorical features shall be divided according to winner and loser as opposed to team colors.
Therefore, each of the 4 categorical columns (barring the target class) will be changed from listing the team to first achieve said trait to either “1” if the team was the winner and “0” if the team was the loser. (i.e. if the value of a row in the “firstBaron” is “Blue”, and the value of the “win” column is “Blue” then the value in the new column shall be “1”, if the “win” value was “Red” then the new value shall be “0”).

Figure 3.9 displays the increase percentage between the winning and losing teams according to the achievement of the particular task of the column. We can learn from it that the “firstTower” achievement is the feature that has the most detrimental in the question of which team will be the winner, it shows that over 70% of the games we won by the team that was able to topple the first enemy turret in the game.

Below, in figure 3.10 we can see a visualization of the data. Which, again, showcases the “firstTower” column importance.



```
winnerFirstBlood:          0.6
loserFirstBlood:           0.4
increase percent:          52.75 %

winnerFirstTower:          0.73
loserFirstTower:           0.27
increase percent:          173.75 %

winnerFirstBaron:          0.61
loserFirstBaron:           0.39
increase percent:          56.39 %

winnerFirstDragon:         0.64
loserFirstDragon:          0.36
increase percent:          74.62 %
```

figure 3.9

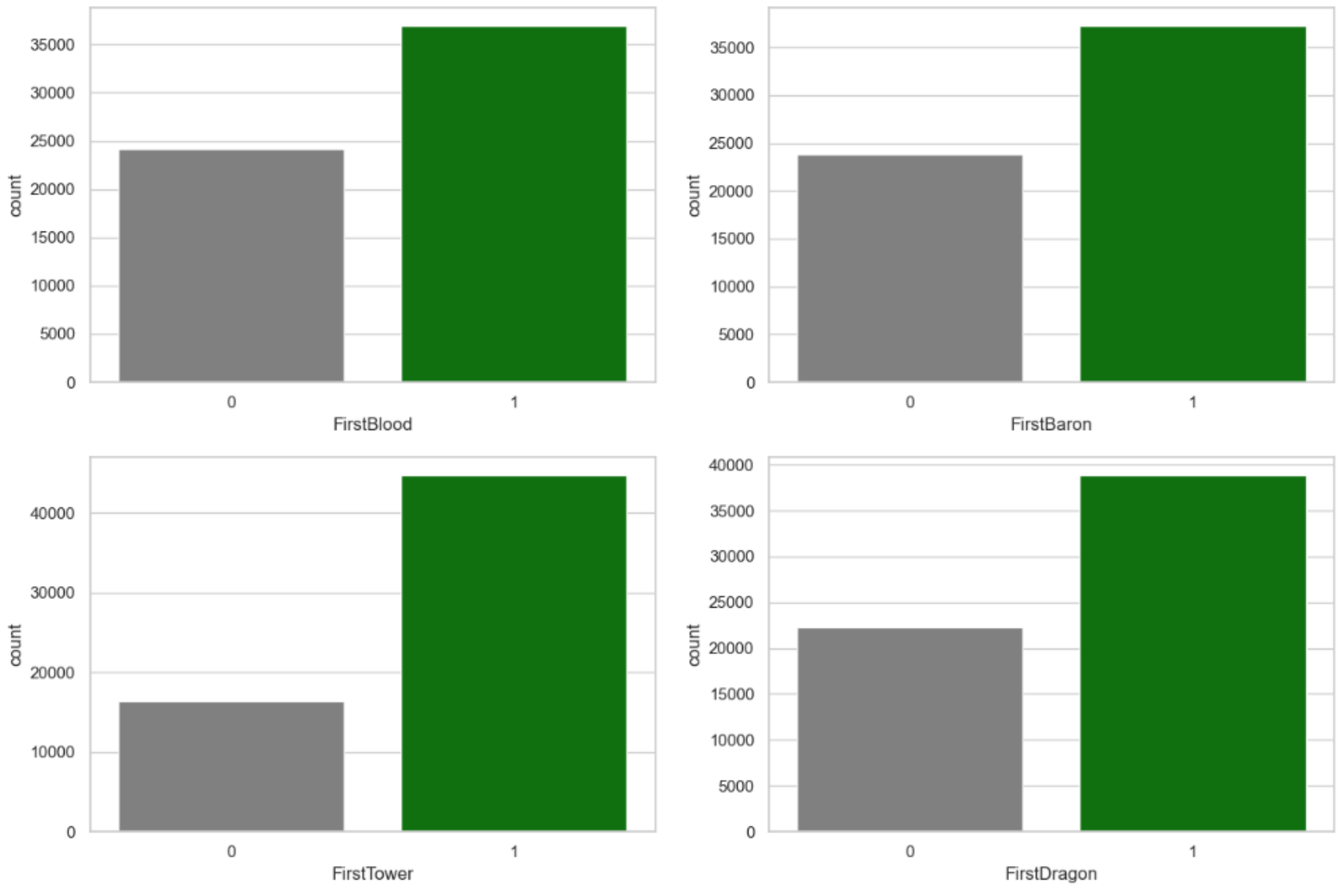


figure 3.10

With these new column we shall take a look at the new correlation map.

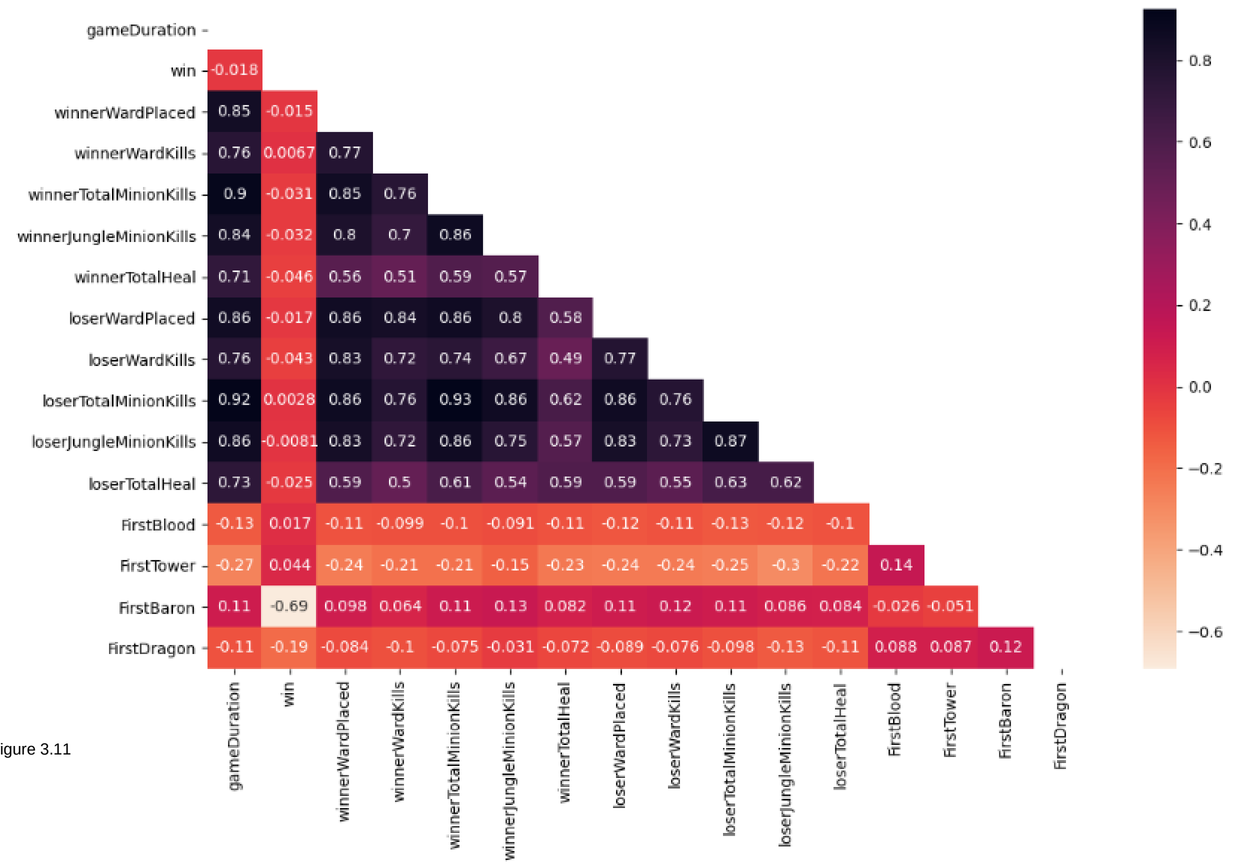


figure 3.11

Unfortunately, it seems that these new columns do not supply us with better correlation with the target class in most cases. Although there is one column that seems very out of place, the ‘FirstBaron’, that has a very high correlation with the target class.

Since the choice in the new columns was not a good one for the most part, I shall go back to the “original” columns (from figure 3.1) in order to figure out how to find better combinations for new columns.

Originally, two of the most pronounced columns where the two jungle minion columns (red and blue).
In order to figure out a way to combine their strength I will try to create a new column consisting of the difference between the values in the ‘jungleMinionKills’. The new column will be called “jungleDiff”.

Two other strong contenders from the original correlation heat map are ‘FirstTower’ and ‘FirstBaron’. I shall create a combine column for these also, called ‘towerBaron’ that will contain either 0, 1, or 2. Zero – if the team that achieved both first tower and first baron was NOT the winning team. Two – if the team that won the game was also the one that achieved both first tower and first baron. And lastly One – if the winning team was able to achieve either the first tower title OR the first baron title.

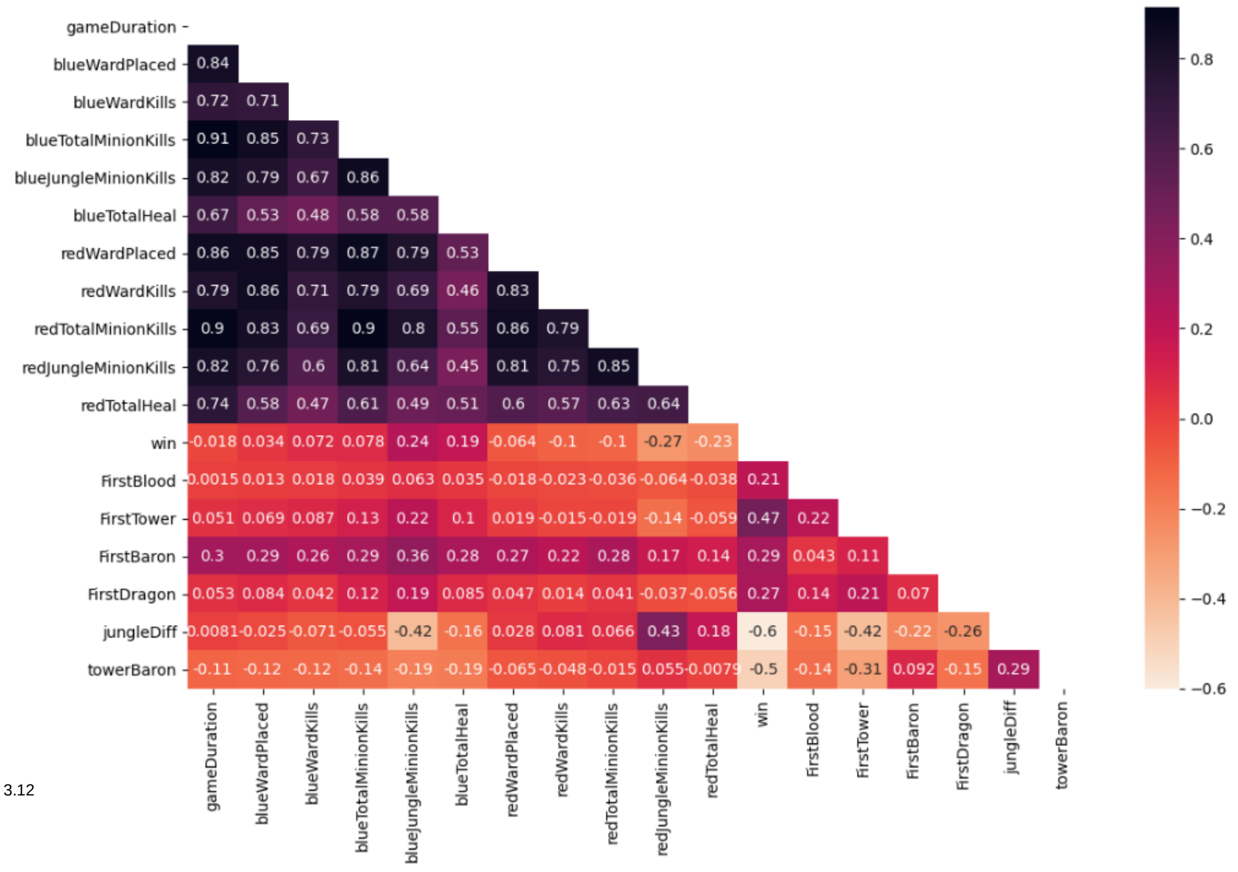


figure 3.12

Looking at this correlation map in figure 3.12 It seems as though the choices of the two new column was a correct one because they both have a relatively high correlation with our target class.

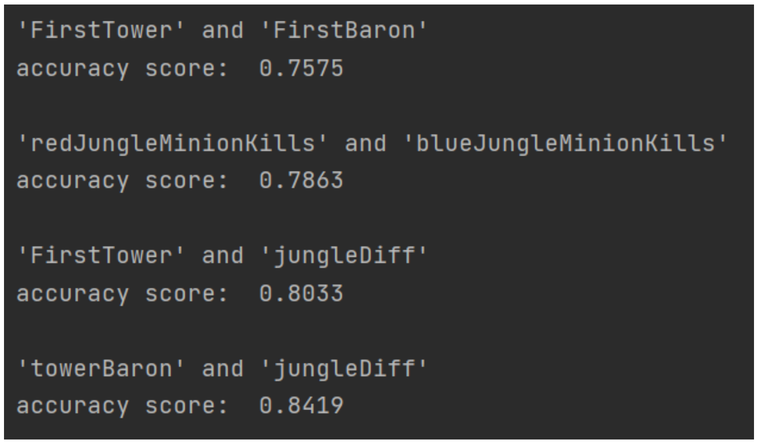
Seeing this, I feel confidant moving on to the Gaussian Naïve Bayes classification model.

4. CLASSIFICATION MODEL

4.1. Gaussian Naïve Bayes

Even though by the end of the previous chapter I have found two features that give the impression of good options for the GNB model, I shall still check the accuracy score of a few options just to make sure that I have chosen well.

Figure 4.1 displays the four pairs that I have checked for accuracy score of the 2 feature GNB classification model.
From thse scores we can tell that the two new columns from chapter 3 are truly the winners in terms of the GNB model.



```
'FirstTower' and 'FirstBaron'
accuracy score:  0.7575

'redJungleMinionKills' and 'blueJungleMinionKills'
accuracy score:  0.7863

'FirstTower' and 'jungleDiff'
accuracy score:  0.8033

'towerBaron' and 'jungleDiff'
accuracy score:  0.8419
```

figure 4.1

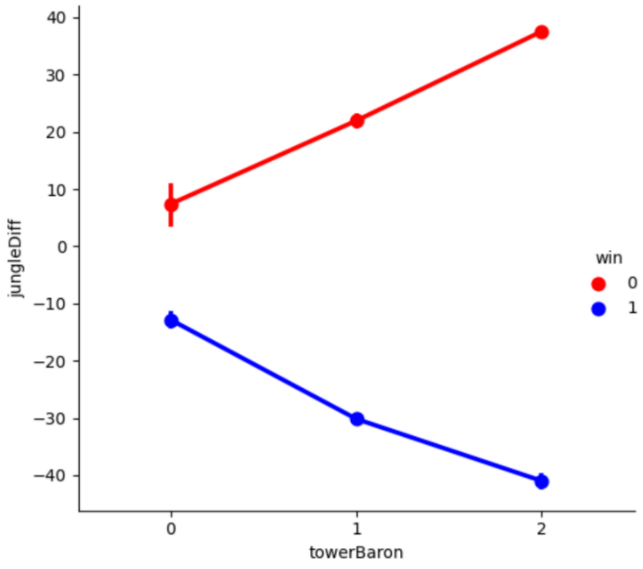


figure 4.2

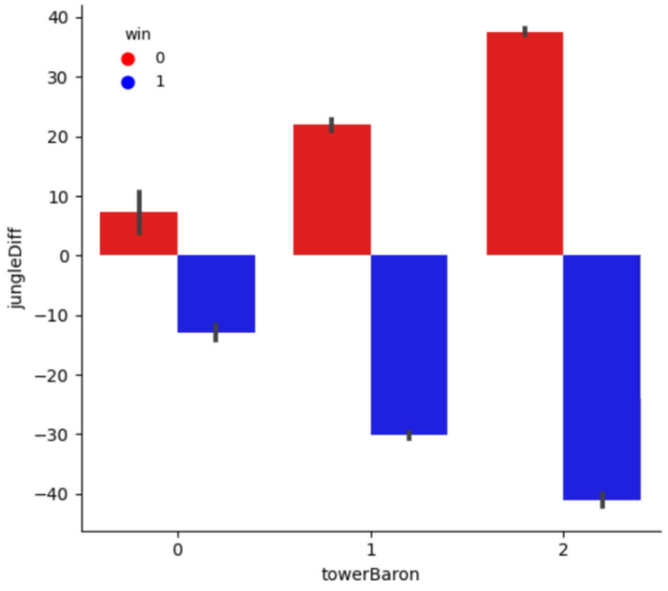


figure 4.3

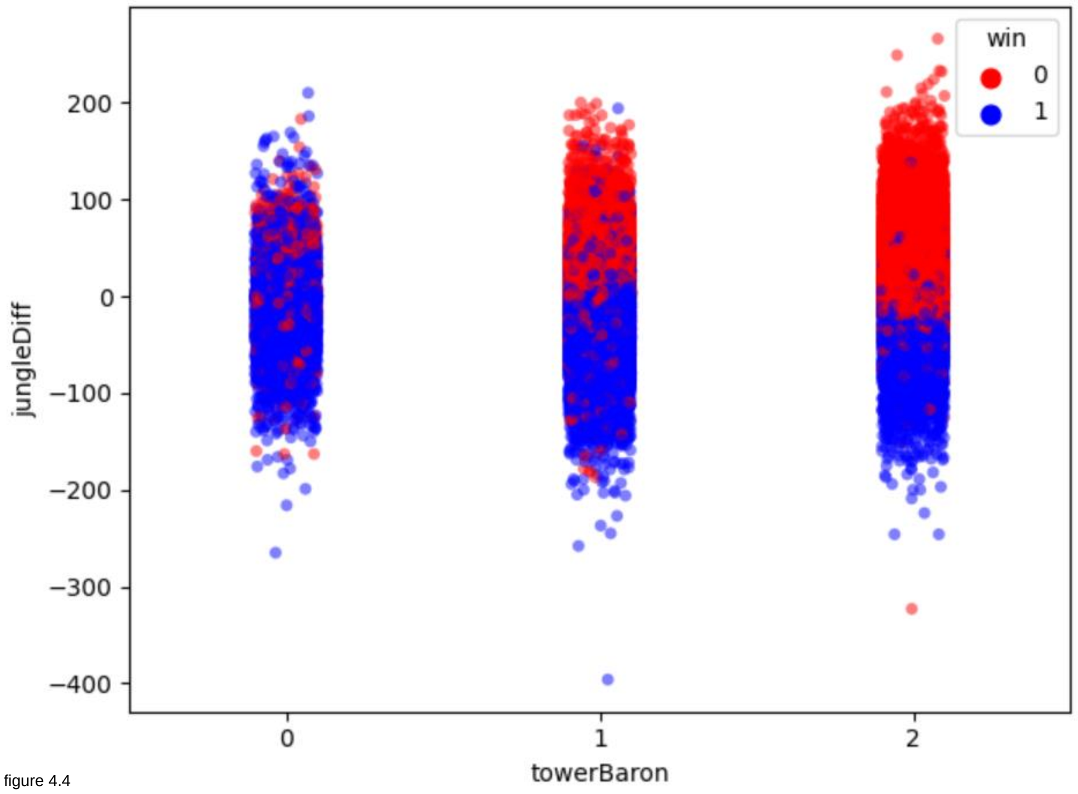


figure 4.4

Figures 4.2, 4.3 and 4.4 display variations of the plots depicting the two columns I shall use for the GNB model.

It is easy to see in all three that a good way to predict if the team will be the victor of the game is if the team has managed to achieve three things:

1. The first to destroy an enemy tower
2. The first to kill an enemy baron
3. The team to kill more jungle minions than opponent

If both of the first achievements were reached, then it is shown in the graphs as the "2" in the X axis.

And the third achievement is shown in the Y axis. The positive values are in favor of the red team (meaning they were able to kill more blue jungle minions) and the negative values are in favor of the blue team.

The Gaussian Naïve Bayes classification model was used with a training set of 80% of the full dataset.

The decision boundaries of the model are shown in figure 4.5 below, along with the scatter of the failed predictions.

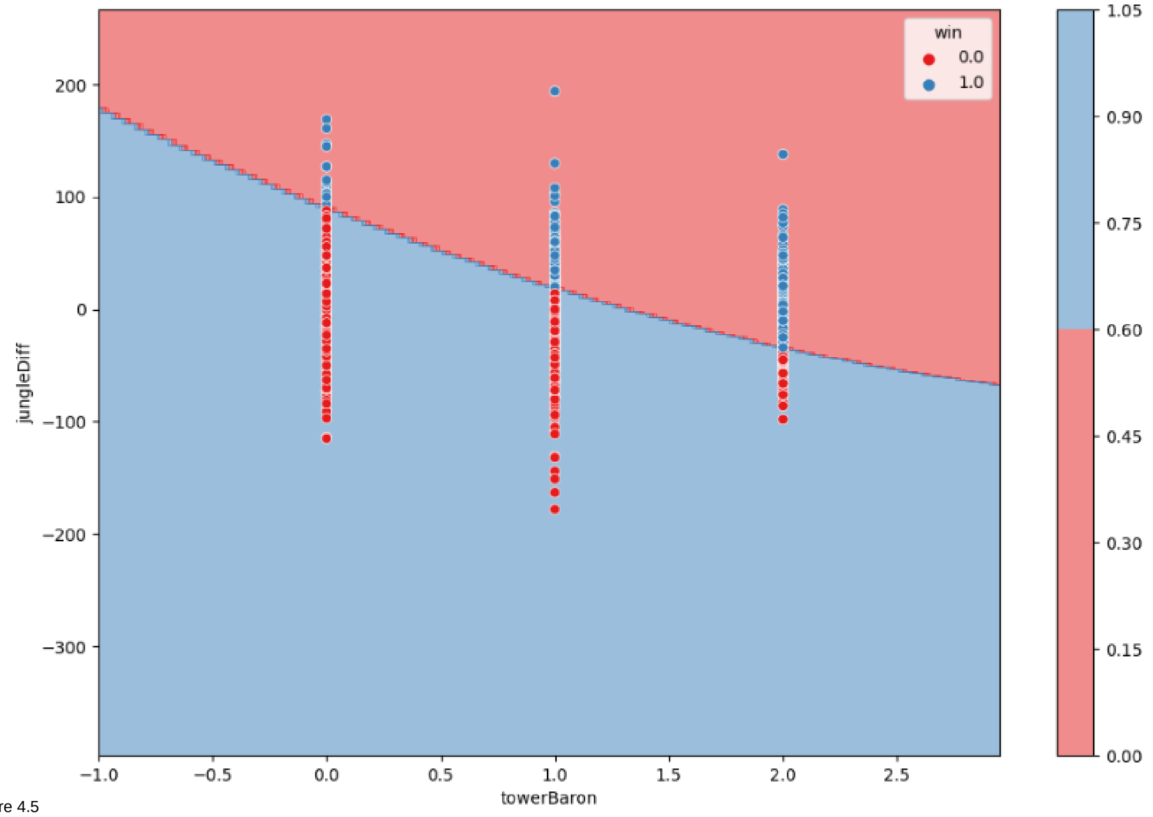


figure 4.5

4.2. Decision Tree

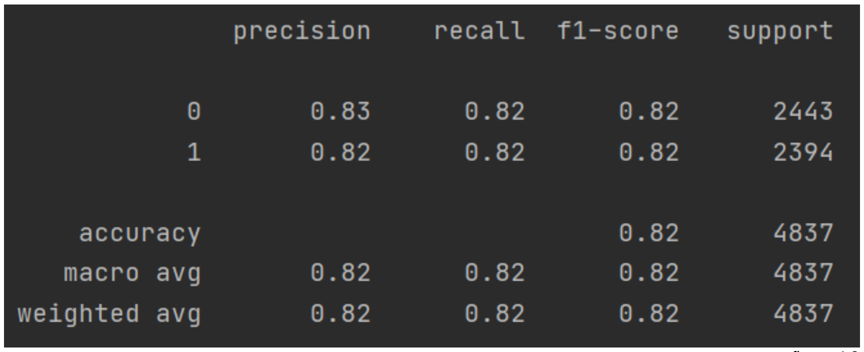
In order to gauge the accuracy of the analysis I have done up to this point, we can take a look at the classification report of two decision trees:

1. Using the original dataset, without the rows that contained missing data.
2. Using the data that I have cleaned and analyzed throughout this project.

4.2.1. Baseline Decision Tree

Using the original data, the tree presents a 82% accuracy score, as displayed in the classification report below. (figure 4.6).

* The Red team is depicted as the ‘0’ and the Blue team as the ‘1’.



```
              precision    recall  f1-score   support

           0       0.83      0.82      0.82      2443
           1       0.82      0.82      0.82      2394

    accuracy                           0.82      4837
   macro avg       0.82      0.82      0.82      4837
weighted avg       0.82      0.82      0.82      4837
```

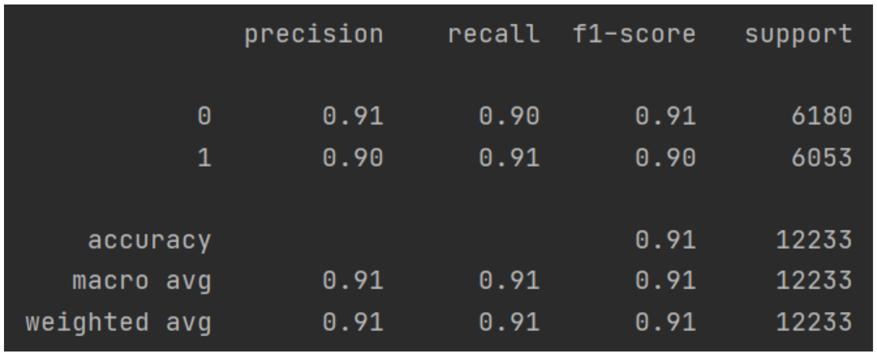
figure 4.6

4.2.2. Manipulated Data Decision Tree

As seen in this classification report (figure 4.7), the accuracy score of the cleaned and analyzed data is indeed better that the score that we got from using the original dataset.

This tells us that the cleaning and changes done to the dataset and its columns has improved the learnability of the dataset.

* The Red team is depicted as the ‘0’ and the Blue team as the ‘1’.



```
              precision    recall  f1-score   support

           0       0.91      0.90      0.91      6180
           1       0.90      0.91      0.90      6053

    accuracy                           0.91     12233
   macro avg       0.91      0.91      0.91     12233
weighted avg       0.91      0.91      0.91     12233
```

figure 4.7

The tree created is very large and impossible to read. Therefore I want to check if it is at all possible to make the tree smaller while preserving the accuracy score as much as possible.
For this reason, I shall look at the permutation importance of the columns, Shown here in figure 4.8.

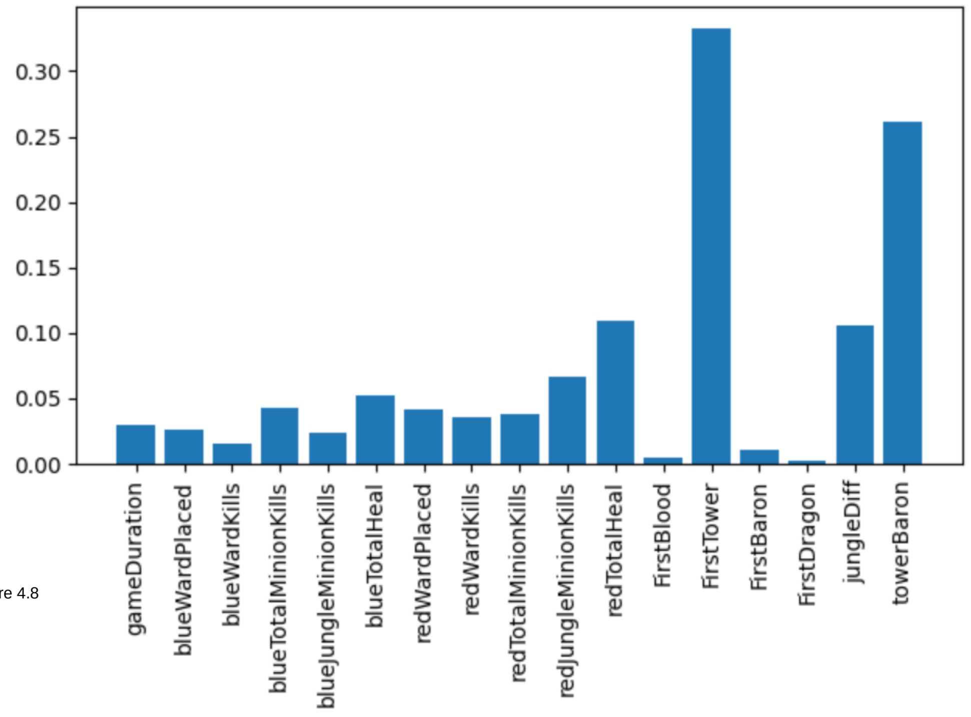
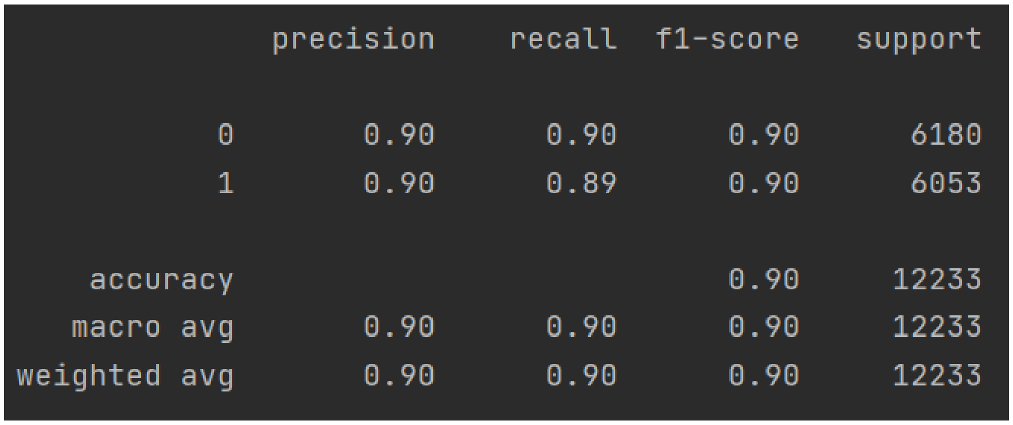


figure 4.8

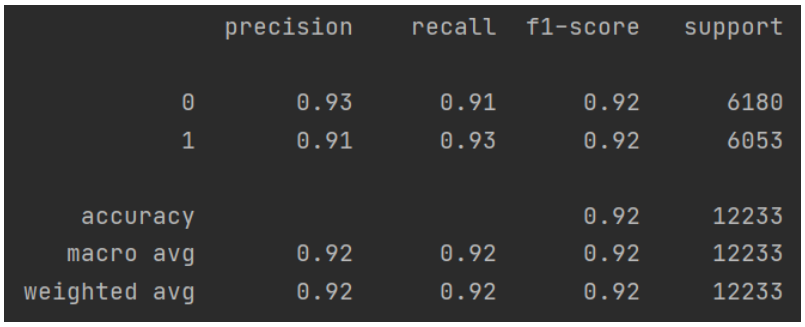
We can see that there are a limited amount of columns that seem very important. Therefore, I will check the accuracy score of the tree made with only 4 of the tallest bars – ‘FirstTower’, ‘towerBaron’, ‘redTotalHeal’, ‘jungleDiff’.


```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90      6180
           1       0.90      0.89      0.90      6053

    accuracy                           0.90     12233
   macro avg       0.90      0.90      0.90     12233
weighted avg       0.90      0.90      0.90     12233
```
figure 4.9

While using only 4 features has definitely impacted the size of the tree, it is still too big to read properly. For this reason I tried limiting the depth of the tree. First I tried limiting the depth to 8, this greatly improved the readability and also actually improved the accuracy score (as seen in figure 4.10).



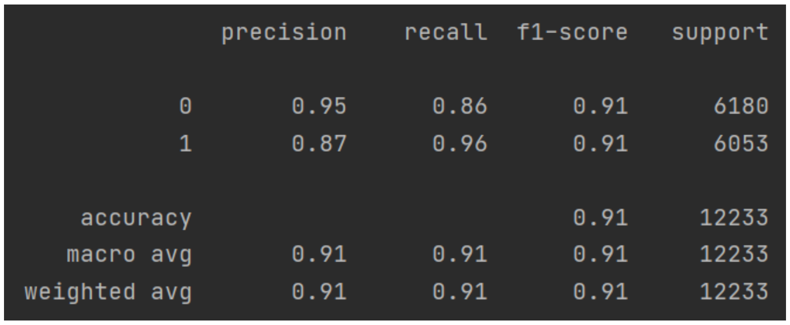
```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      6180
           1       0.91      0.93      0.92      6053

    accuracy                           0.92     12233
   macro avg       0.92      0.92      0.92     12233
weighted avg       0.92      0.92      0.92     12233
```

figure 4.10

Seeing the vast improvements, I was curious if using a smaller depth could make an even better impact. Using a max-depth of 4 lowered the accuracy (figure 4.11), but checking values between 4 and 8, I found that 5 was “the sweet spot” since it both offered an outstanding accuracy score, and also provided a very concise tree that was easy to read (figure 4.12 and 4.13).

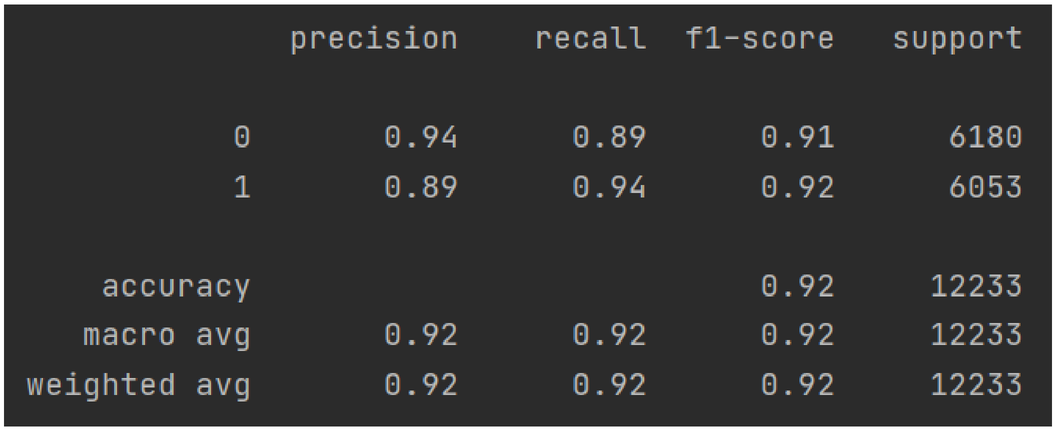


```
              precision    recall  f1-score   support

           0       0.95      0.86      0.91      6180
           1       0.87      0.96      0.91      6053

    accuracy                           0.91     12233
   macro avg       0.91      0.91      0.91     12233
weighted avg       0.91      0.91      0.91     12233
```

figure 4.11



```
              precision    recall  f1-score   support

           0       0.94      0.89      0.91      6180
           1       0.89      0.94      0.92      6053

    accuracy                           0.92     12233
   macro avg       0.92      0.92      0.92     12233
weighted avg       0.92      0.92      0.92     12233
```

figure 4.12

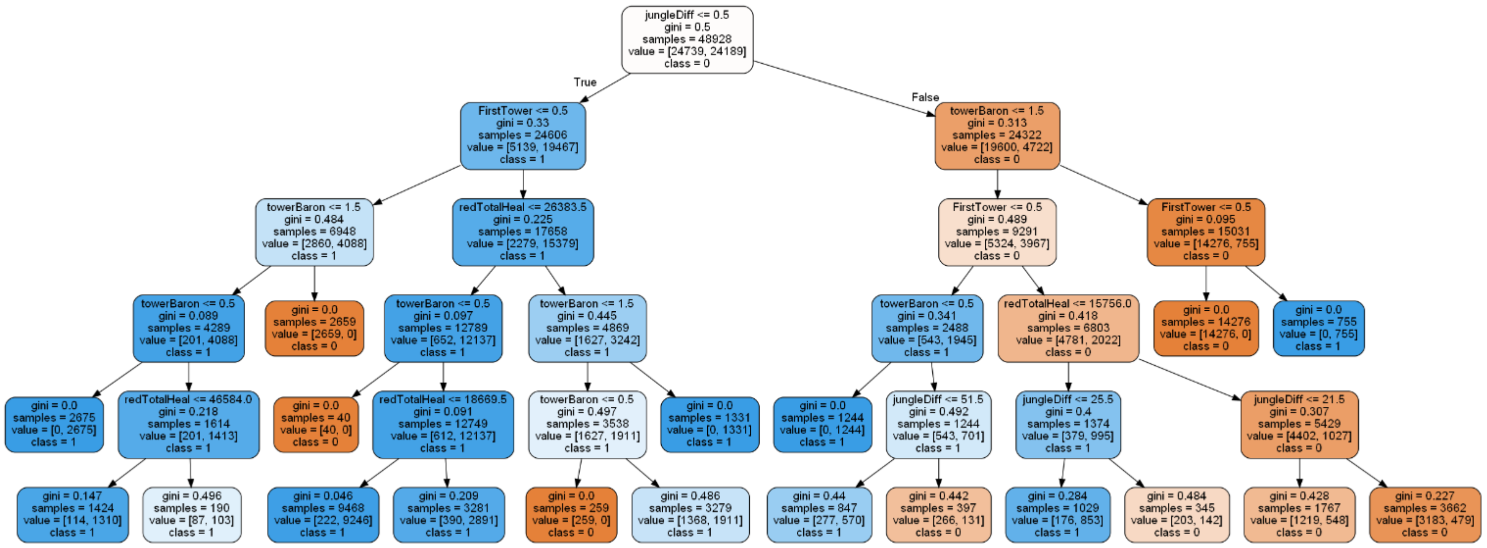

jungleDiff <= 0.5
gini = 0.5
samples = 48928
value = [24739, 24189]
class = 0
True
False
FirstTower <= 0.5
gini = 0.33
samples = 24606
value = [5139, 19467]
class = 1
towerBaron <= 1.5
gini = 0.313
samples = 24322
value = [19600, 4722]
class = 0
towerBaron <= 1.5
gini = 0.484
samples = 6948
value = [2860, 4088]
class = 1
redTotalHeal <= 26383.5
gini = 0.225
samples = 17658
value = [2279, 15379]
class = 1
FirstTower <= 0.5
gini = 0.489
samples = 9291
value = [5324, 3967]
class = 0
FirstTower <= 0.5
gini = 0.095
samples = 15031
value = [14276, 755]
class = 0
towerBaron <= 0.5
gini = 0.089
samples = 4289
value = [201, 4088]
class = 1
gini = 0.0
samples = 2659
value = [2659, 0]
class = 0
towerBaron <= 0.5
gini = 0.097
samples = 12789
value = [652, 12137]
class = 1
towerBaron <= 1.5
gini = 0.445
samples = 4869
value = [1627, 3242]
class = 1
towerBaron <= 0.5
gini = 0.341
samples = 2488
value = [543, 1945]
class = 1
redTotalHeal <= 15756.0
gini = 0.418
samples = 6803
value = [4781, 2022]
class = 0
gini = 0.0
samples = 14276
value = [14276, 0]
class = 0
gini = 0.0
samples = 755
value = [0, 755]
class = 1
gini = 0.0
samples = 2675
value = [0, 2675]
class = 1
redTotalHeal <= 46584.0
gini = 0.218
samples = 1614
value = [201, 1413]
class = 1
gini = 0.0
samples = 40
value = [40, 0]
class = 0
redTotalHeal <= 18669.5
gini = 0.091
samples = 12749
value = [612, 12137]
class = 1
towerBaron <= 0.5
gini = 0.497
samples = 3538
value = [1627, 1911]
class = 1
gini = 0.0
samples = 1331
value = [0, 1331]
class = 1
gini = 0.0
samples = 1244
value = [0, 1244]
class = 1
jungleDiff <= 51.5
gini = 0.492
samples = 1244
value = [543, 701]
class = 1
jungleDiff <= 25.5
gini = 0.4
samples = 1374
value = [379, 995]
class = 1
jungleDiff <= 21.5
gini = 0.307
samples = 5429
value = [4402, 1027]
class = 0
gini = 0.147
samples = 1424
value = [114, 1310]
class = 1
gini = 0.496
samples = 190
value = [87, 103]
class = 1
gini = 0.046
samples = 9468
value = [222, 9246]
class = 1
gini = 0.209
samples = 3281
value = [390, 2891]
class = 1
gini = 0.0
samples = 259
value = [259, 0]
class = 0
gini = 0.486
samples = 3279
value = [1368, 1911]
class = 1
gini = 0.44
samples = 847
value = [277, 570]
class = 1
gini = 0.442
samples = 397
value = [266, 131]
class = 0
gini = 0.284
samples = 1029
value = [176, 853]
class = 1
gini = 0.484
samples = 345
value = [203, 142]
class = 0
gini = 0.428
samples = 1767
value = [1219, 548]
class = 0
gini = 0.227
samples = 3662
value = [3183, 479]
class = 0

figure 4.13

5. SUMMARY

This report presents that a strong relationship exists between a few of the criteria of the League of Legends game achievements and the determination of the winning team.
I have found throughout this project that the team in the game that has attained three specific successes has an extremely high chance of winning the game.
The three feats any team should strive for in the LoL game are:

1. Be the first team to destroy an enemy turret
2. Be the first team to kill an enemy baron
3. Be the team to kill more jungle minions than the enemy

A team that has accomplished all three is almost guaranteed to be the victor.

During the exploration of this dataset and the analyzing of it, I have come across some difficulties, mostly in the ‘cleaning’ phase and when choosing the best new columns to create from the existing values. Although I did have some struggle, I feel that I have achieved the desired results, especially considering that the new data that I have created and the cleaning that I have done has indeed improved the learnability of the dataset and increased the accuracy scores of both the Gaussian Naïve Bayes classification model and the Decision Tree classification model.

In conclusion, while I was not familiar with the game prior to this assignment, I just might consider playing a time or two even if just to see if I can experience my findings for real.

* note: The presentation video is uploaded to my google drive account at this address:
https://drive.google.com/file/d/1PCyjMqHgpon8gunI1TWXUIHnPJG-TzET/view?usp=sharing