League of Legends Classification

YiZhi Zhang, Lai Wei

31.12.2018

STAT 441

INTRODUCTION

League of Legends (LOL) is a famous online MOBA game that has lasted for 10 years. 10 players are having a 5 vs 5 competition during every game until one team has destroy another team's base. During every game, each player chooses a champion, killing minions, monsters, and enemies' champions to earn golds. Then purchase items to assistant to destroy enemy's base. Not only the champion user selected are significant, golds, towers, but do the cooperation between teammates contributes significantly to the final result.

Motivation and introduction of the problem

Every year, LOL worldwide professional competition will be held around the world. Numbers of professional teams fight for their dreams. As the audience, people are significantly expected their favorite team to win. Consequently, many people are willing to develop a method to predict if one team can win during each match. At the same time, since there are many aspect such as gold and kills that may influence during the game, athletes and other LOL players are also interested in how those aspects affect the result of each match.

Consequently, as the main goal of this project, we will focus on the relationship between the amount of golds earned by each five positions (Top, Jungle, Middle, Attack Damage Carry (ADC) and Support) for both blue team (consider as ally) and red team (consider as enemy) and the result of the match. More specifically, we will use different statistical classification models to classify if the blue team in a match will win or lose according to the amount of gold earned by all ten players at the 10th minute, the 20th minute, and finally the 30th minute. And analyses how amounts of gold during each match influence the result of that match.

Data

The original data are collected from Kaggle (url:https://www.kaggle.com/chuckephron/leagueoflegends#LeagueofLegends.csv), which is distributed by Chuck Ephron.

This data has total 57 columns and 7620 observations that including all professional competitions from 2015 to 2018. The data has the following aspects:

1.1 Address: website address the data is scraped from.

1.2 League: League or Tournament the match took place in.

1.3 Year: Year the match took place in.

1.4 Season: Spring or Summer depending on which half of the year the match took place in.

1.5 Type: Season, Playoffs, Regional, or International match.

1.6 Team Tag for both sides: (blueTeamTag, redTeamTag).

1.7 gamelength: the length of the match.

1.8 Gold: the gold earned by all ten players in each minute, the total amount of gold in both teams, and the gold difference.

1.9 Champions: The champions that are banned before the game and the champions used by players.

1.10 Players: the names (ID) of all players.

1.11 Kills: the killer, victim, assistants, time, and location for each kill.

1.12 Dragons/Barons/Heralds: the time and the object time slayed by each side, for dragon the types of dragon are also included.

1.13 Towers/Inhibitors: the time the construction destroyed and the position of it.

This data set almost include all the information that can be collected during each match. However, there are some problems that prevent to be used for classification. For this reason, we clean this data set to make sure it works for all classification models.

Data Preprocessing

The script used is “DataClean.py”

1.1 Data cleaning: string

Since R can easily handle string easily by applying as.numeric or as.factor as needed, string features do not need to be deleted or extremely modified here.

For all string features, the script shows that they are proper enough. Thus, we do not modify them.

1.2 Data cleaning: array

Array values in this data do not have a fixed length. Thus the data we remained is the length of those arrays, and the first element of those arrays

Golds:

Calculate the golds for all two teams for all five positions for 10min, 20min, and 30 min.

If the games ends before 30min, we will set the gold to be the amount at the end of gold.

1.3 Counting Legendary Monsters: Dragon

Count the first dragon slayed for both sides, and the total number of dragons for both sides.

If one side does not slay any dragon, record the time as the end of game.

There are elements dragon in 2017 and 2018, we ignore them here for now.

1.4 Counting Legendary Monsters: Baron/Herald

Count the first baron slayed for both sides.

1.5 Counting Towers: Tower/Inhibitor

Count the first tower destroyed for both sides, and the total number of towers/Inhibitors for both sides.

If one side does not destroy any tower/Inhibitor, record the time as the end of the game.

1.6 Counting Kills

Count the first kill for both sides, and the total number of kills for both sides.

If one side does not kill, record the time as the end of the game.

Finally, we drop all address, champions banned, rResult, Year, Season, (b/r)TeamTag, League, Type and name of players since they are not important or hard to clean. The result data named “LOL.csv”, contains only checked string, floating points, and integers. For response, integer “0” indicates that blue team (Ally) loses that match, and “1” indicates that ally wins that match. See “DataClean.pdf” for more details.

There is a significant characteristic in both the original dataset and the new dataset, which is that data is highly correlated. The reason is related the actual property of LOL, for example, team who get a Baron will be given a positive buff that they can easily destroy towers, which can also influence their golds. If we only focus on gold only, team which gain more golds at the beginning means all members in that time have more gold to purchase items, which means they can get other benefits easier than the enemy to earn more gold later. Consequently, a champion who gain more golds may be easier to gain more rapidly later. There are also some potential dependencies between different positions, which will be discussed and tested later.

Methodology

First, we will using Linear Discriminant Analysis (LDA) to classify the whole dataset, we get the following result:

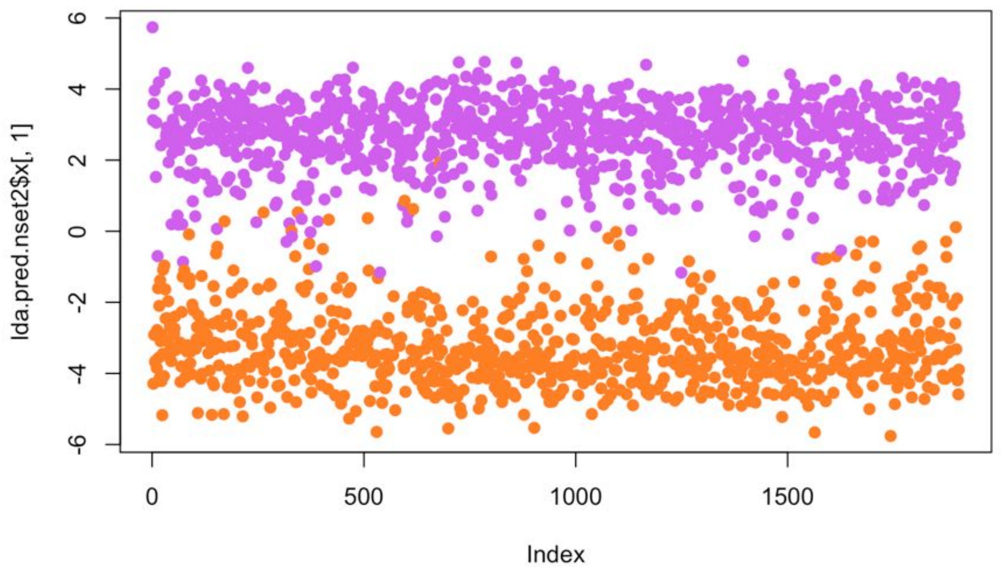
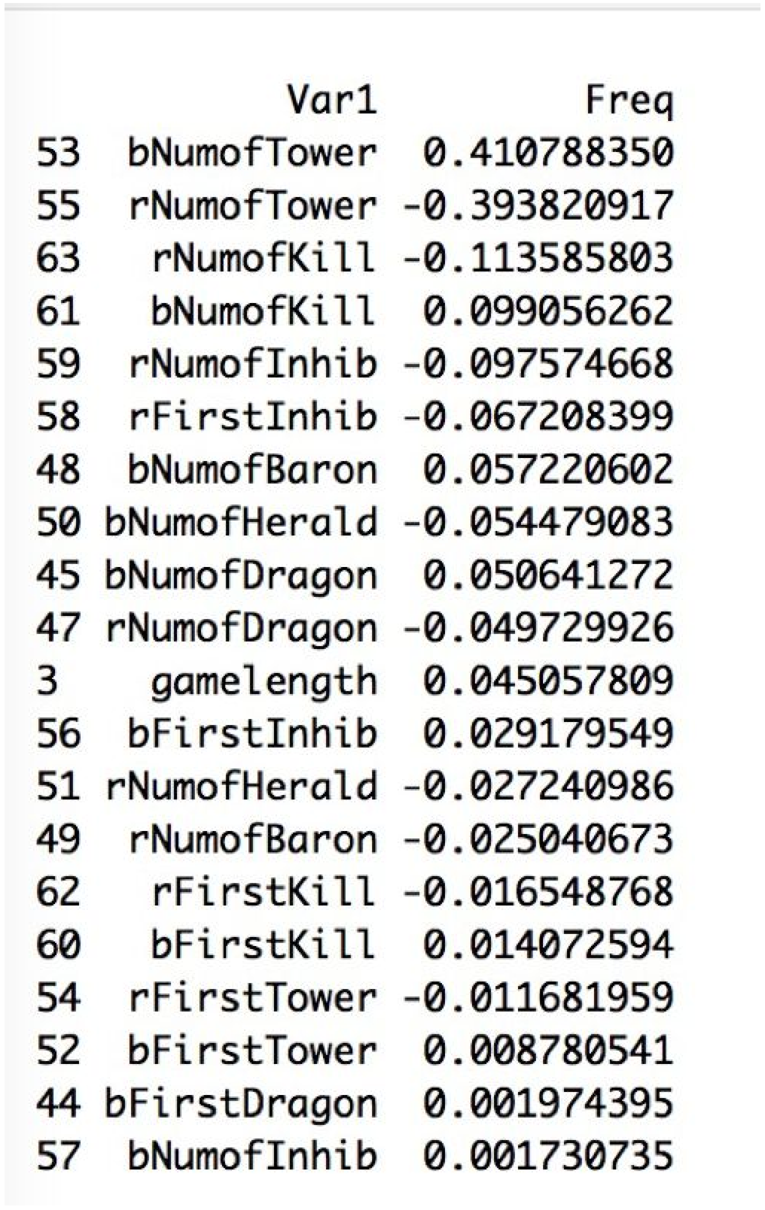


Figure 1: LDA Visualization for whole data set

Error rate:

	Error Rate
Training Set Error	0.01181102
Validation Set Error	0.01207349
Testing Set Error	0.009973753

With all the explanatory variables, LDA has already been a perfect split with 1% test error rate. Look into the importance of explanatory variable:



```
                 Var1         Freq
53   bNumofTower   0.410788350
55   rNumofTower  -0.393820917
63    rNumofKill  -0.113585803
61    bNumofKill   0.099056262
59   rNumofInhib  -0.097574668
58   rFirstInhib  -0.067208399
48   bNumofBaron   0.057220602
50 bNumofHerald   -0.054479083
45 bNumofDragon    0.050641272
47 rNumofDragon   -0.049729926
3     gamelength   0.045057809
56   bFirstInhib   0.029179549
51 rNumofHerald   -0.027240986
49   rNumofBaron  -0.025040673
62    rFirstKill  -0.016548768
60    bFirstKill   0.014072594
54   rFirstTower  -0.011681959
52   bFirstTower   0.008780541
44 bFirstDragon    0.001974395
57   bNumofInhib   0.001730735
```

Figure 2: Significance for explanatory variables

Examining the top 20 explanatory variables, we can see that the explanatory variables named "number of ***" are especially important. It's reasonable, since these explanatory variables are too strong as they are the data of the game when the whole game is over. For example, if a team has more kills when the game is over, we can make an intuitive guess that that team wins the game.

And in terms of prediction, these explanatory variables do not help the analysis. We actually cannot achieve them until the end of the game. To achieve the model for prediction, we should drop these explanatory variables. (also the "game length")

For the rest of the project, we will drop all explanatory except golds (30 columns total). To achieve the goal of this project, which is to discover the relationship between golds at some time and the result of the match. The datasets we used later on are:

Response and the amount of gold at the 10th minute for all ten players.

Response and the amount of gold at the 10th minute and 20th minute for all ten players.

Response and the amount of gold at the 10th minute, 20th minute, and 30th minute for all ten players.

According to the property of LOL, we first make a hypothesis that the amount of gold for each player in ally team has a positive relationship to the probability to win, while the amount of gold for each player in the enemy team has a negative relationship instead. We also make the hypothesis that the amount of gold in the 30th minute is a significant feature for the result of a match. In other words, classifiers built and trained with gold data will work properly.

Linear Discriminant Analysis (LDA)

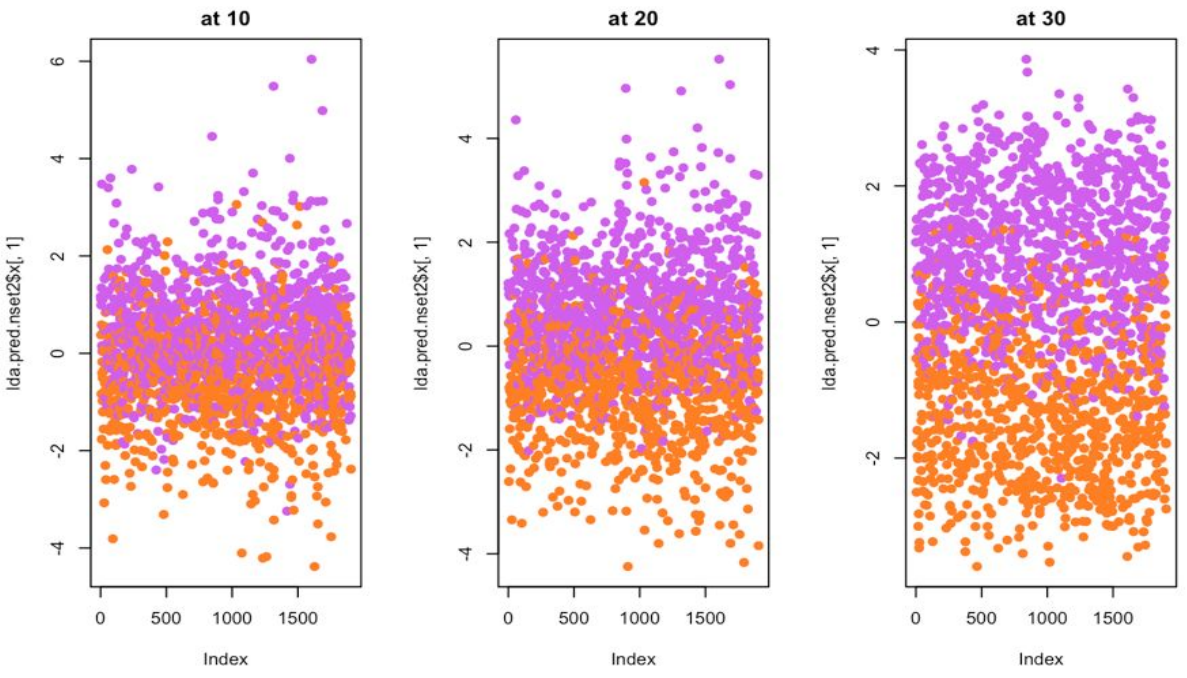


Figure 3: LDA visualization for 10th, 20th, 30th minute gold data

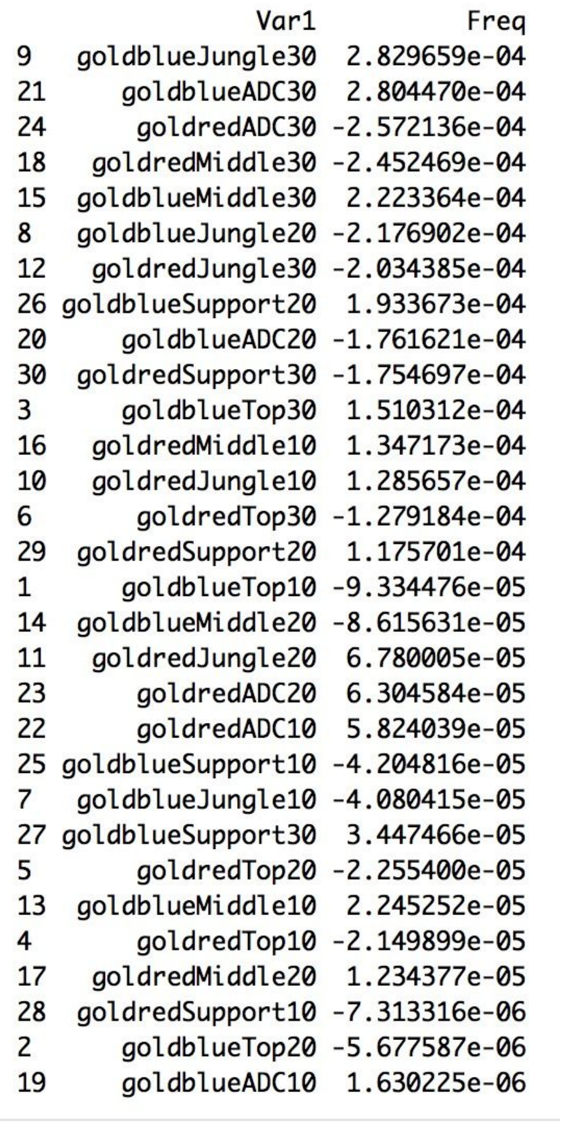
Error rate:

	10th minute	20th minute	30th minute
Training Error	0.3149606	0.2233596	0.1244094
Validation Error	0.3070866	0.2215223	0.1312336
Test Error	0.3275591	0.2283465	0.1270341

As we can see from the first scatter-plot graph, which is the classification visualization for classifying ally-winning matches (purple) and ally-defeating matches (orange) where line $y = 0$ is a projection of decision boundary. It is obvious that those three models, especially the third one, perform properly on classification on their respective dataset. It is also obvious that each additional column that contains the later gold data plays a significant role on classifying.

According to the errors table, LDA classifiers for this case do not appear a overfitting problem since all three type errors are similar. As the result shown, LDA classifiers expect to have approximately 68%, 77%, and 87% accuracy rate if we provide the 10th, 20th, and 30th minute of gold data respectively.

Let’s see the summary for the LDA classifier (sorted) using all gold data:



```
                 Var1          Freq
9    goldblueJungle30  2.829659e-04
21      goldblueADC30  2.804470e-04
24       goldredADC30 -2.572136e-04
18    goldredMiddle30 -2.452469e-04
15   goldblueMiddle30  2.223364e-04
8    goldblueJungle20 -2.176902e-04
12    goldredJungle30 -2.034385e-04
26 goldblueSupport20  1.933673e-04
20      goldblueADC20 -1.761621e-04
30   goldredSupport30 -1.754697e-04
3       goldblueTop30  1.510312e-04
16    goldredMiddle10  1.347173e-04
10    goldredJungle10  1.285657e-04
6        goldredTop30 -1.279184e-04
29   goldredSupport20  1.175701e-04
1       goldblueTop10 -9.334476e-05
14   goldblueMiddle20 -8.615631e-05
11    goldredJungle20  6.780005e-05
23       goldredADC20  6.304584e-05
22       goldredADC10  5.824039e-05
25 goldblueSupport10 -4.204816e-05
7    goldblueJungle10 -4.080415e-05
27 goldblueSupport30  3.447466e-05
5        goldredTop20 -2.255400e-05
13   goldblueMiddle10  2.245252e-05
4        goldredTop10 -2.149899e-05
17    goldredMiddle20  1.234377e-05
28   goldredSupport10 -7.313316e-06
2       goldblueTop20 -5.677587e-06
19      goldblueADC10  1.630225e-06
```

Figure 4: Significance for explanatory variables

We can see that for all ten players, the amounts of gold in the 30th minute have the highest coefficients of LDA, which means those features contribute more for classification than other features.

Quadratic Discriminant Analysis (QDA)

Error rate:

	10th minute	20th minute	30th minute
Training Error	0.3312336	0.2380577	0.1320210
Validation Error	0.3191601	0.2524934	0.1522310
Test Error	0.3517060	0.2530184	0.1522310

According to the errors table, QDA classifiers show a small trend of overfitting that all validation errors and test errors are 2% higher than training error. As the result shown, QDA classifiers expect to have approximately 65%, 75%, and 85% accuracy rate if we provide the 10th, 20th, and 30th minute of gold data respectively.

Compare with LDA, QDA classifiers work slightly worse. One possible reason is that the assumption for LDA, which is that both classifiers have same correlations for each explanatory variable respectively, is meaningful in this case, while the problem for QDA, which is that QDA tends to overfit, negatively influence the classification results.

Logistic Regression (LG)

Error rate:

	10th minute	20th minute	30th minute
Training Error	0.3165354	0.2233596	0.1225722

Validation Error	0.3039370	0.2251969	0.1307087
Test Error	0.3270341	0.2246719	0.1291339

According to the errors table, LG classifiers for this case do not appear a overfitting problem since all three type errors are similar. As the result shown, LG classifiers expect to have approximately 68%, 77%, and 87% accuracy rate if we provide the 10th, 20th, and 30th minute of gold data respectively, which is almost the same as LDA classifiers.

Logistic Regression with LASSO (LGL)

Let’s see the CV-1SE result for three datasets first:

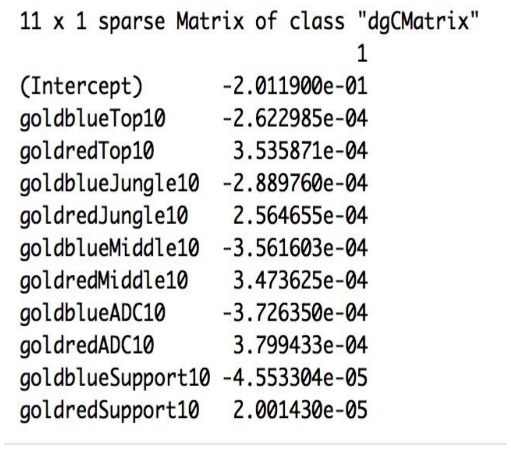
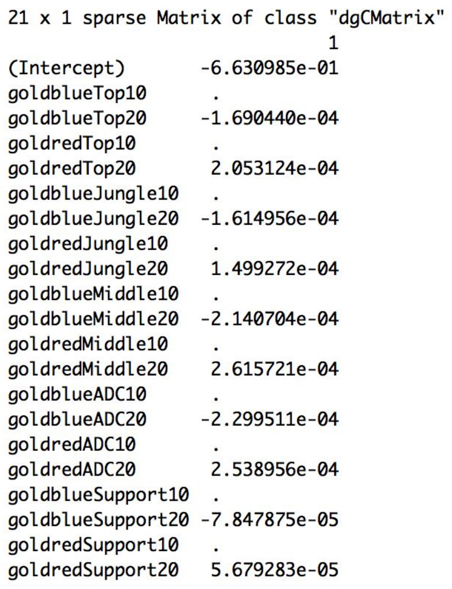
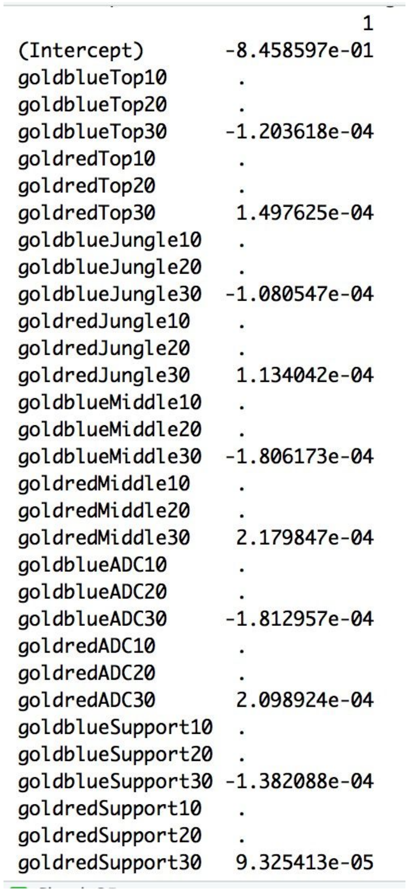

```
11 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)       -2.011900e-01
goldblueTop10     -2.622985e-04
goldredTop10       3.535871e-04
goldblueJungle10  -2.889760e-04
goldredJungle10    2.564655e-04
goldblueMiddle10  -3.561603e-04
goldredMiddle10    3.473625e-04
goldblueADC10     -3.726350e-04
goldredADC10       3.799433e-04
goldblueSupport10 -4.553304e-05
goldredSupport10   2.001430e-05
```

Figure 5: Coefficients of explanatory variables at 10 min



```
21 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)       -6.630985e-01
goldblueTop10      .
goldblueTop20     -1.690440e-04
goldredTop10       .
goldredTop20       2.053124e-04
goldblueJungle10   .
goldblueJungle20  -1.614956e-04
goldredJungle10    .
goldredJungle20    1.499272e-04
goldblueMiddle10   .
goldblueMiddle20  -2.140704e-04
goldredMiddle10    .
goldredMiddle20    2.615721e-04
goldblueADC10      .
goldblueADC20     -2.299511e-04
goldredADC10       .
goldredADC20       2.538956e-04
goldblueSupport10  .
goldblueSupport20 -7.847875e-05
goldredSupport10   .
goldredSupport20   5.679283e-05
```

Figure 6: Coefficients of explanatory variables at 20 min

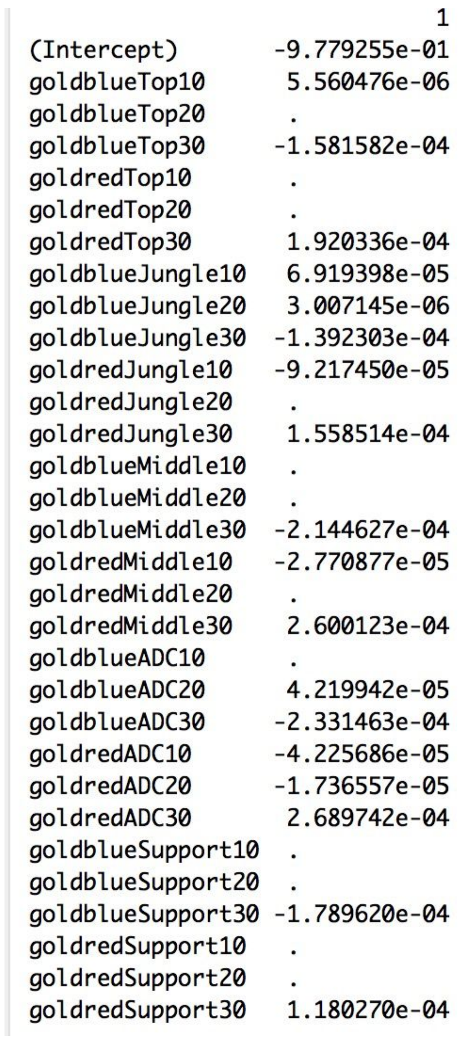


```
                              1
(Intercept)       -8.458597e-01
goldblueTop10      .
goldblueTop20      .
goldblueTop30     -1.203618e-04
goldredTop10       .
goldredTop20       .
goldredTop30       1.497625e-04
goldblueJungle10   .
goldblueJungle20   .
goldblueJungle30  -1.080547e-04
goldredJungle10    .
goldredJungle20    .
goldredJungle30    1.134042e-04
goldblueMiddle10   .
goldblueMiddle20   .
goldblueMiddle30  -1.806173e-04
goldredMiddle10    .
goldredMiddle20    .
goldredMiddle30    2.179847e-04
goldblueADC10      .
goldblueADC20      .
goldblueADC30     -1.812957e-04
goldredADC10       .
goldredADC20       .
goldredADC30       2.098924e-04
goldblueSupport10  .
goldblueSupport20  .
goldblueSupport30 -1.382088e-04
goldredSupport10   .
goldredSupport20   .
goldredSupport30   9.325413e-05
```

Figure 7: Coefficients of explanatory variables at 10 min

We can see that if CV 1-SE is used, all previous gold amount that is before the latest one will be regularized to 0. Which means only the latest amount of gold will be used.

Compare to the CV-Min Method in the 30th minute dataset:



```
                             1
(Intercept)       -9.779255e-01
goldblueTop10      5.560476e-06
goldblueTop20      .
goldblueTop30     -1.581582e-04
goldredTop10       .
goldredTop20       .
goldredTop30       1.920336e-04
goldblueJungle10   6.919398e-05
goldblueJungle20   3.007145e-06
goldblueJungle30  -1.392303e-04
goldredJungle10   -9.217450e-05
goldredJungle20    .
goldredJungle30    1.558514e-04
goldblueMiddle10   .
goldblueMiddle20   .
goldblueMiddle30  -2.144627e-04
goldredMiddle10   -2.770877e-05
goldredMiddle20    .
goldredMiddle30    2.600123e-04
goldblueADC10      .
goldblueADC20      4.219942e-05
goldblueADC30     -2.331463e-04
goldredADC10      -4.225686e-05
goldredADC20      -1.736557e-05
goldredADC30       2.689742e-04
goldblueSupport10  .
goldblueSupport20  .
goldblueSupport30 -1.789620e-04
goldredSupport10   .
goldredSupport20   .
goldredSupport30   1.180270e-04
```

Figure 8: Coefficients of explanatory variables at 30 min using lambda min

We can see CV-Min error penalize less heavily on the magnitude of coefficient. Thus some 20th minute data that are significant get their coefficients back.

Error rate using λ as the cv min:

	10th minute	20th minute	30th minute

Training Error	0.3162730	0.2241470	0.1270341
Validation Error	0.3154856	0.2225722	0.1338583
Test Error	0.3275591	0.2272966	0.1291339

Error rate using λ as the cv 1 standard error:

	10th minute	20th minute	30th minute
Training Error	0.3162730	0.2238845	0.1278215
Validation Error	0.3154856	0.2225722	0.1312336
Test Error	0.3275591	0.2262467	0.1291339

Applying LASSO regularization seems not improve or worsen the result at all in this case, which means the features remained after regularization dominate enough for classification. In this case, we can see that the gold data in the 30th minute dominates the result.

Naive Bayes (NB)

Error rate without using PCA rotation:

	10th minute	20th minute	30th minute
Training Error	0.3265092	0.2343832	0.1517060
Validation Error	0.3107612	0.2362205	0.1627297

Test Error	0.3406824	0.2467192	0.1601050

Error rate with using PCA rotation:

	10th minute	20th minute	30th minute
Training Error	0.3215223	0.2341207	0.1354331
Validation Error	0.3107612	0.2362205	0.1386325
Test Error	0.3328084	0.2309711	0.1406824

From two Error tables, even though we can see that using PCA rotation gets a better job than not, both results are both worse than LDA or LR especially for the 30th minute dataset. The major as mentioned before is that NB assume each explanatory variable are independent to each other, which is not a proper assumption for these datasets. Hence in this case GB is not a good choice.

Logistic Generalized Additive Model (GAM)

Error rate:

	10th minute	20th minute	30th minute
Training Error	0.3144357	0.2207349	0.1223097
Validation Error	0.3091864	0.2325459	0.1364829
Test Error	0.3265092	0.2251969	0.1343832

According to the errors table, GAM classifiers for this case do not appear a overfitting problem since all three type errors are similar. As the result shown, GAM classifiers expect to have approximately 68%, 77%, and 86% accuracy rate if we provide the 10th, 20th, and 30th minute of gold data respectively, which is almost the same as LDA classifiers. However, GAM model has a problem, which is that the executing time is significantly longer than LR.

Single Classification Tree (SCT)

Parameter Tuning: use 1-SE method here, we select cp = 0.012, 0.0068 0.0074 for three datasets respectively.

Let's see the tree visualization:

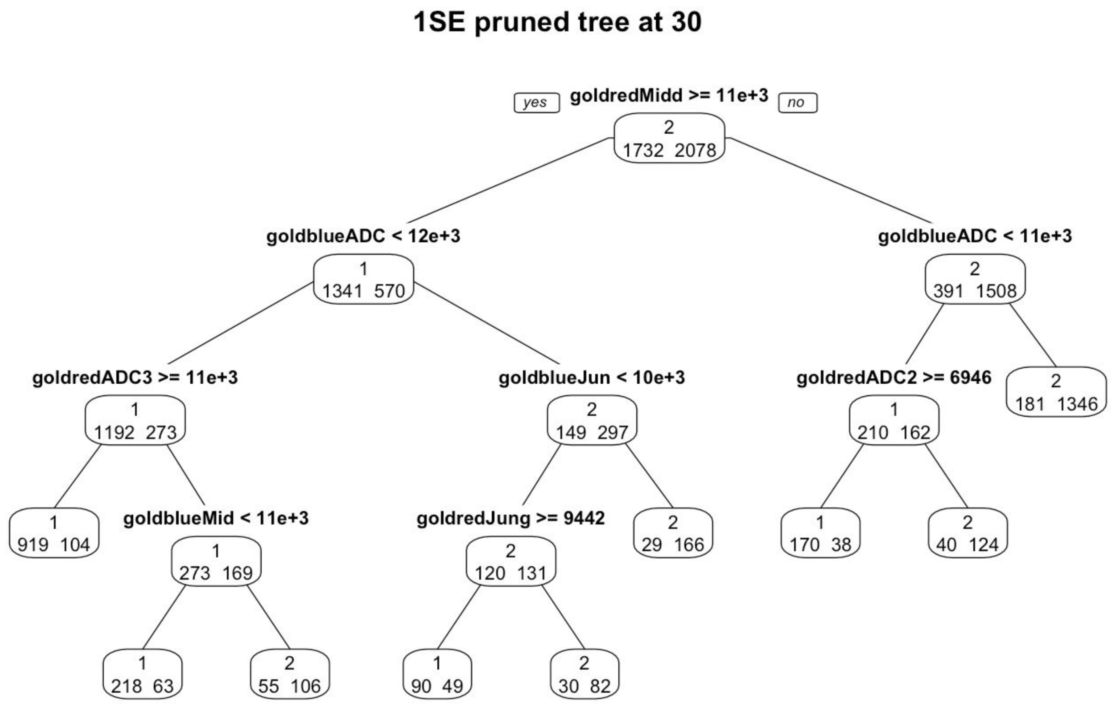


Figure 9: Pruned classification tree at 30 min

From the classification tree, we can see all intermediate node split the data most according to the amount of gold in the 30th minute, some of them are 20th minute, which means the results of the match majorly depend on the 30th minute gold data.

Furthermore, we can see that gold for mid and ADC are the most significant explanatory variables, while Top and Support are not that important.

Error rate:

	10th minute	20th minute	30th minute
Training Error	0.3412073	0.2475066	0.1545932
Validation Error	0.3711286	0.2776903	0.1790026
Test Error	0.3979003	0.2708661	0.1784777

According to the error table, the single classification tree model does not perform well and it has the overfitting issue. The single classification tree is very easy to interpret, as shown on the above plot. However, it is a quite variable model which results in severe overfit. Meanwhile, it cannot handle the potential additive structure (for example, in this data set, the change of gold between different times could be of some importance)

These drawbacks are the reasons why we proceed to use random forest.

Random Forest (RF)

We tune the tree number ntree for the three data sets. It turns out to be similar: 500 trees and 1500 trees will result in larger OOB error compared to 1000 trees. So we use ntree = 1000 for all three data sets. According to the Gini index as a measure of influence, the middle and top's gold still are the most important feature to measure the winning rate.

Error rate:

	10th minute	20th minute	30th minute
Training Error	0.3488189	0.2346457	0.1417323
Validation Error	0	0	0
Test Error	0.3422572	0.2377953	0.1517060

From the error table, we can see that the random forest model fit the validation set to zero error! But the test error shows that the prediction is acceptable and better than the single classification tree.

Gradient Boosting Model (GBM)

We choose the tune parameter to be the recommended value.

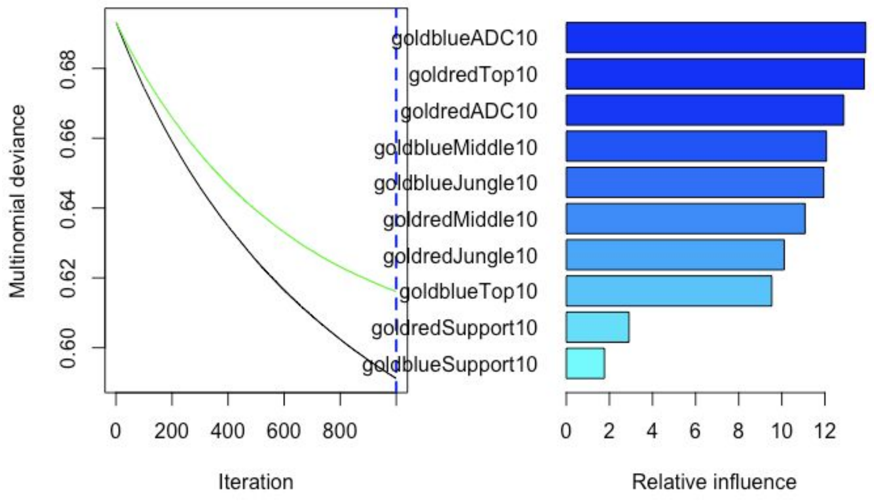


Figure 10: Influence hist plot at 10 min

For the 10 mins data, we can see that the gold is of similar importance except for the support. But for the 20 mins data, the influence starts to vary.

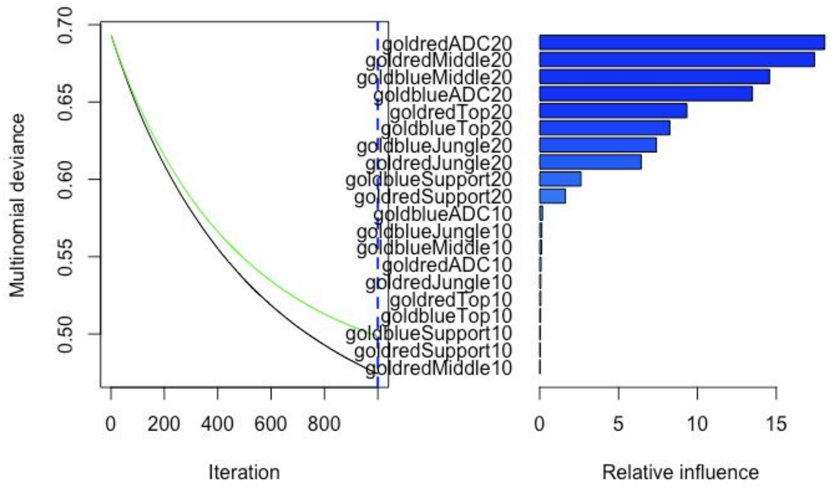


Figure 11: Influence hist plot at 20 min

The data of the first 10 mins tend to be of no use, and the gold of ADC and middle position becomes more important than that of jungle and top position. And for the game at 30 mins, we find an interesting fact that unlike what it is at 10mins or 20mins, the support positions' gold has high influence on the probability of the victory.

	10th minute	20th minute	30th minute
Training Error	0.2897638	0.2023622	0.1123360
Validation Error	0.3238845	0.2278215	0.1401575
Test Error	0.3391076	0.2393701	0.1333333

Gradient boosting does a really good job on all three data sets. But it might be quite consuming to tune the gbm and predict using the tuned model.

Support Vector Machine (SVM)

As we using validation set to tune parameters gamma and cost, we get the following result:

Gamma	cost
1e-08	1e+07

Hence, we will use the tuned parameters to train SVM models.
Error rate:

	10th minute	20th minute	30th minute
Training Error	0.3178478	0.2230472	0.1236220

Validation Error	0.3107612	0.2267717	0.1291339
Test Error	0.3254593	0.2262467	0.1328084

According to the errors table, SVM classifiers for this case do not appear a overfitting problem since all three type errors are similar. As the result shown, SVM classifiers expect to have approximately 67%, 77%, and 85% accuracy rate if we provide the 10th, 20th, and 30th minute of gold data respectively, which is almost the same as the classifiers before.

Total Result

In total, we validate 11 different models on the data sets.

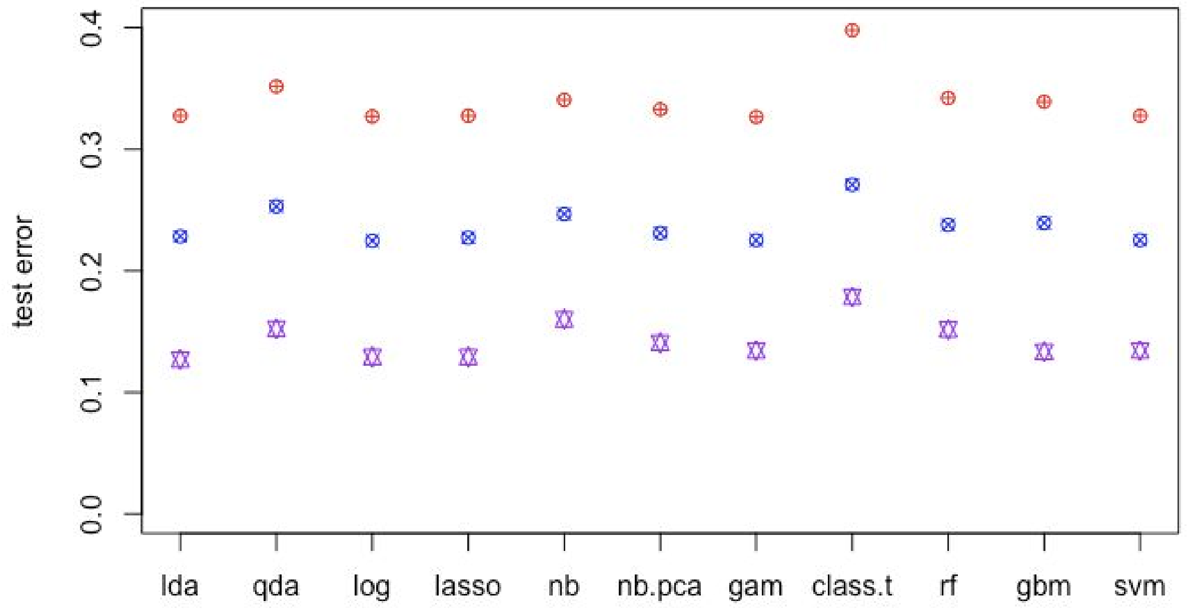


Figure 12: Test error of different models

The figure shows the test errors of different models on the three data sets: at 10min, 20min, and 30min. It is clear that as time grows, we achieve more data, so all the models are able to predict better. And from the variable selection analysis on the models, we notice that the data of the latest gold are always of larger importance.

Future Thoughts

This report only focuses on the relationship between gold and the match result. To make a more accurate model to predict the result of a match, we can use more aspects such as kills and monster slayed.

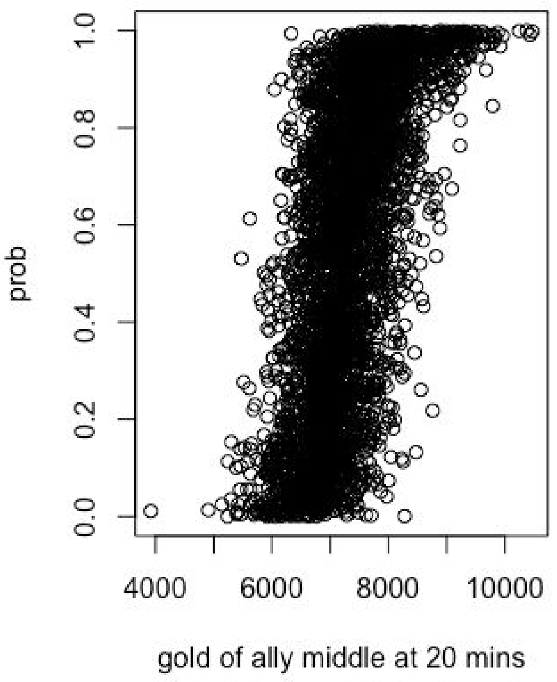
In addition, we can also analysis if champions that have higher winning rates contribute to a professional match, or how champions that are strong at the beginning period influence together with the amount of gold. That’s also some topic that we can get through with this dataset.

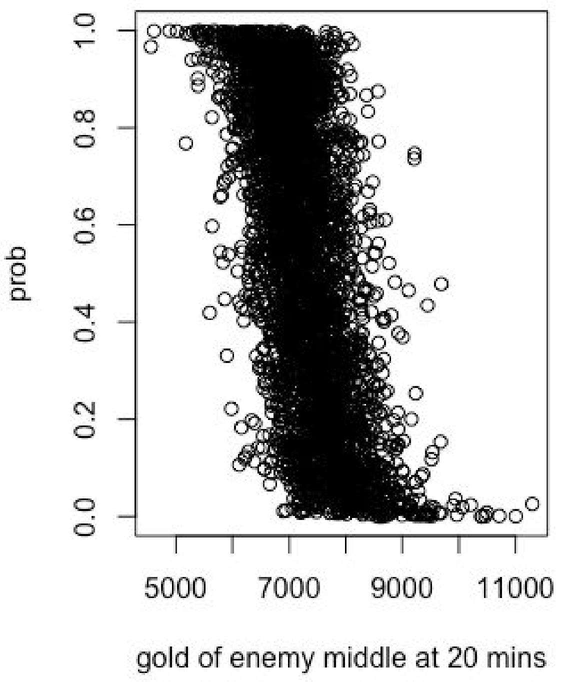
Conclusion

Compare with those models, we can see that some models (such as LDA, LR w/o LASSO, GAM, and SVM) perform similar and better than other models. Since in reality, not only if the team can win but also how much is the winning rate are things that people are always interested in, we conclude that Logistic Regression with or without LASSO Regularization are two of the most proper choices for classifying if the blue team in a match can win depends on the amount of gold at the beginning of the match.

According to the graph shown in some models, we can see that if we use all gold data, the amount of gold in the 30th minute dominates the classification result. If we get rid of the gold in the 30th minute, we can see that the 20th minute dominates the classification result. Hence, the amount of gold has an increasing influence on the result of a match from the 10th minute to 30th minute.

As we can see in the Logistic Regression Coefficients, the amounts of gold in ally team all have positive relationship with ally’s winning rate, while the amounts of gold in enemy team all have negative relationship with ally’s winning rate, which match our hypothesis at the beginning.





Most of the model get a prediction accuracy around 67%, 77%, and 87% respect to 10th minute, 20th minute, and the 30th minute. On the one hand, this result shows that there is a strong relationship between the gold earned and the winning rate. On the other hand, gold advantage does not lead to a 100% winning probability, which means a team that experience an inferior situation still have the probability to win. That is exactly the spirit of sports and competitions.