Predicting the Win Rate and Duration of Game LOL in Real Time

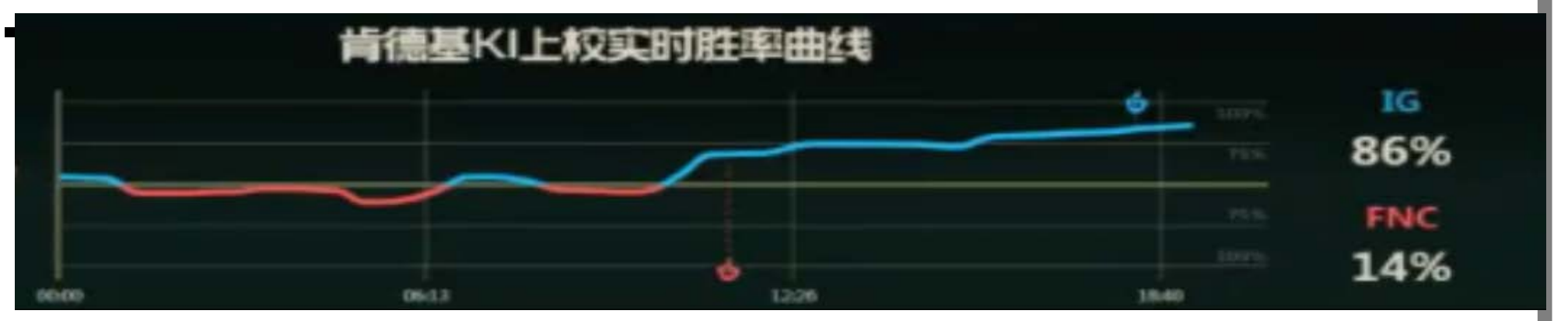
Based on Classification and Linear Regression

Xiaohan Xu, Yujie Ren, Directed by Dr. Zelenberg
Beijing Institute of Technology

ILLINOIS INSTITUTE
OF TECHNOLOGY

Introduction

▪**League of Legends (abbreviated LoL) is a multiplayer online battle arena video game. The goal is usually to destroy the opposing team's "nexus", a structure that lies at the heart of a base protected by defensive structures. There are many factors can influence the win rate and duration of game, such as tower, difference of gold, dragon ,inhibitor, etc. And the win rate and duration are what people are always concerned about.**



▪**In our project, we will predict win rate and duration of game in real time by some predictors in the game based on the dataset "(LoL) League of Legends Ranked Games". And the techniques are classification and linear regression respectively. At the same time, we get our optimization model by different method of fitting, forward stepwise selection, lasso etc.**

- ❑ **Technique: Linear regression ,Logistic regression, LDA, KNN.**
- ❑ **Optimizing method: Forward stepwise selection, Lasso**

Theory and formula of linear regression

Multiple Linear Regression Model is used to regress and predict each match's game duration.

- **Linear Model:**

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

(Where β_i means coefficient, X means predictors, and ε means the error.)

- **Estimating coefficient:**

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p.$$

- **Figuring out whether X affects Y:** do the null hypothesis $H_0 : \beta_1 = 0$ calculate the t-value and p-value:

$$t = \frac{\hat{\beta}_1 - 0}{\mathrm{SE}(\hat{\beta}_1)},$$

- **F-statistic:**

null hypothesis: $H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$

$$F = \frac{(\mathrm{TSS} - \mathrm{RSS})/p}{\mathrm{RSS}/(n-p-1)};$$

- **Assessing the Accuracy of the Model: We define:**

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

$$\mathrm{TSS} = \sum (y_i - \bar{y})^2$$

$$\mathrm{RSE} = \sqrt{\frac{1}{n-p-1} RSS} = \sqrt{\frac{1}{n-p-1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

$$R^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

RSE is used to evaluate the deviation of our model;
R^2 is used to evaluate our model's fitness to the data.

Theory about Classification

- ***Logistic Regression***
  - **Getting estimate β by get maximum of likelihood function**

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}.$$

$$\ell(\beta_0, \beta_1) = \prod_{i: y_i = 1} p(x_i) \prod_{i': y_{i'} = 0} (1 - p(x_{i'})).$$

- ***LDA(Linear Discriminant Analysis)***
  - ***Bayes' theorem***

$$\Pr(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

  - **Compare *discriminant function***

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

▪***KNN(K-Nearest Neighbors)***

- **By Calculating the number of the nearest points of every class respectively.**

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

▪***Forward Stepwise Selection***

- ***Aim: reduce the number of predictors***

▪***The Lasso***

- ***Aim: reduce the variance.***
- **By minimizing the quality of right equation.**

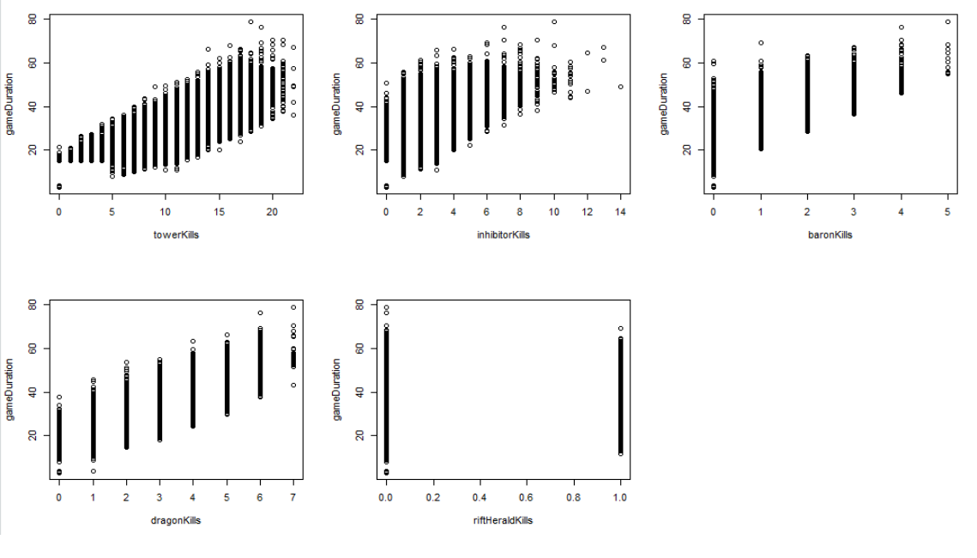
$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \mathrm{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$$

Implementing Linear Regression Model to regress and predict the game duration

1. Process the Dataset

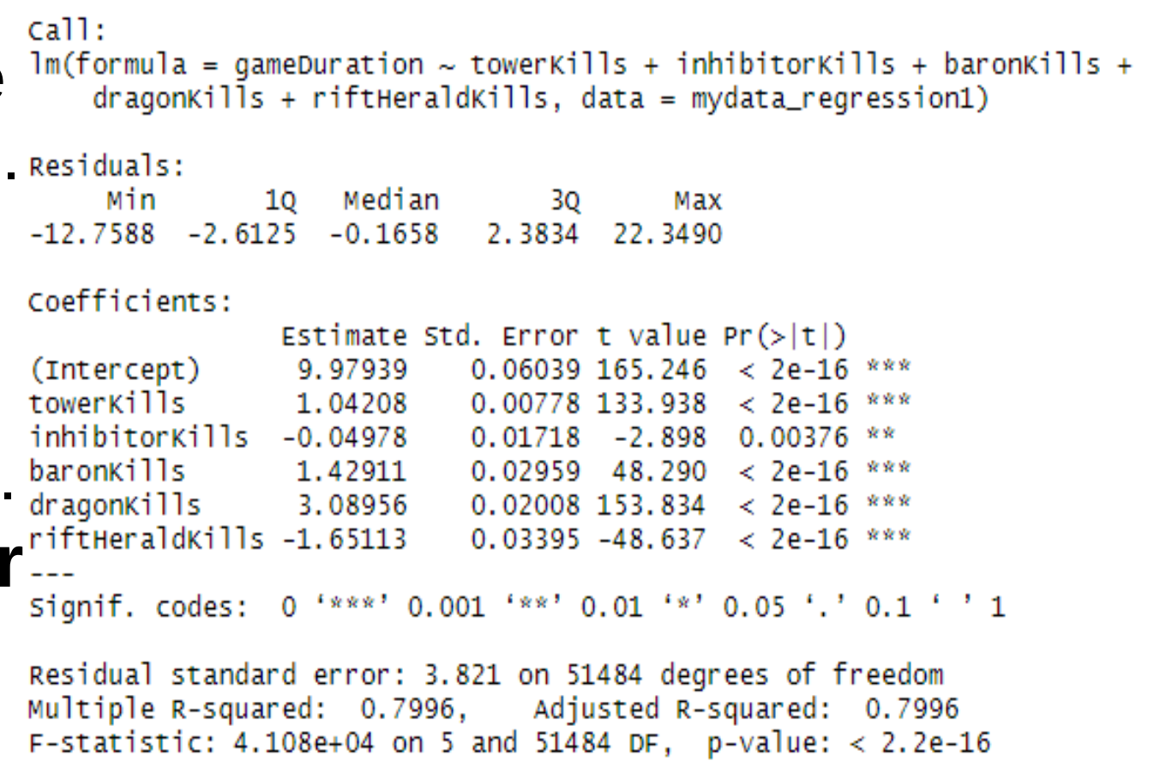
There are several works to do to deal with the dataset:

① Delete the useless or missing observations.
② Select predictors roughly.
③ Change yes-no question into 0-1 value
④ Combine two-teams' variables into one.
⑤ Change Response's unit from seconds to minut



2. Select predictors using stepwise

There are 5 variables left, and we will select the best ones to predict, using the method **stepwise**. (In R, stepwise can be done by a function called "stepAIC()", which is from MASS package.) The result of this step is on the right side. We can conclude that all of the variables are chosen.



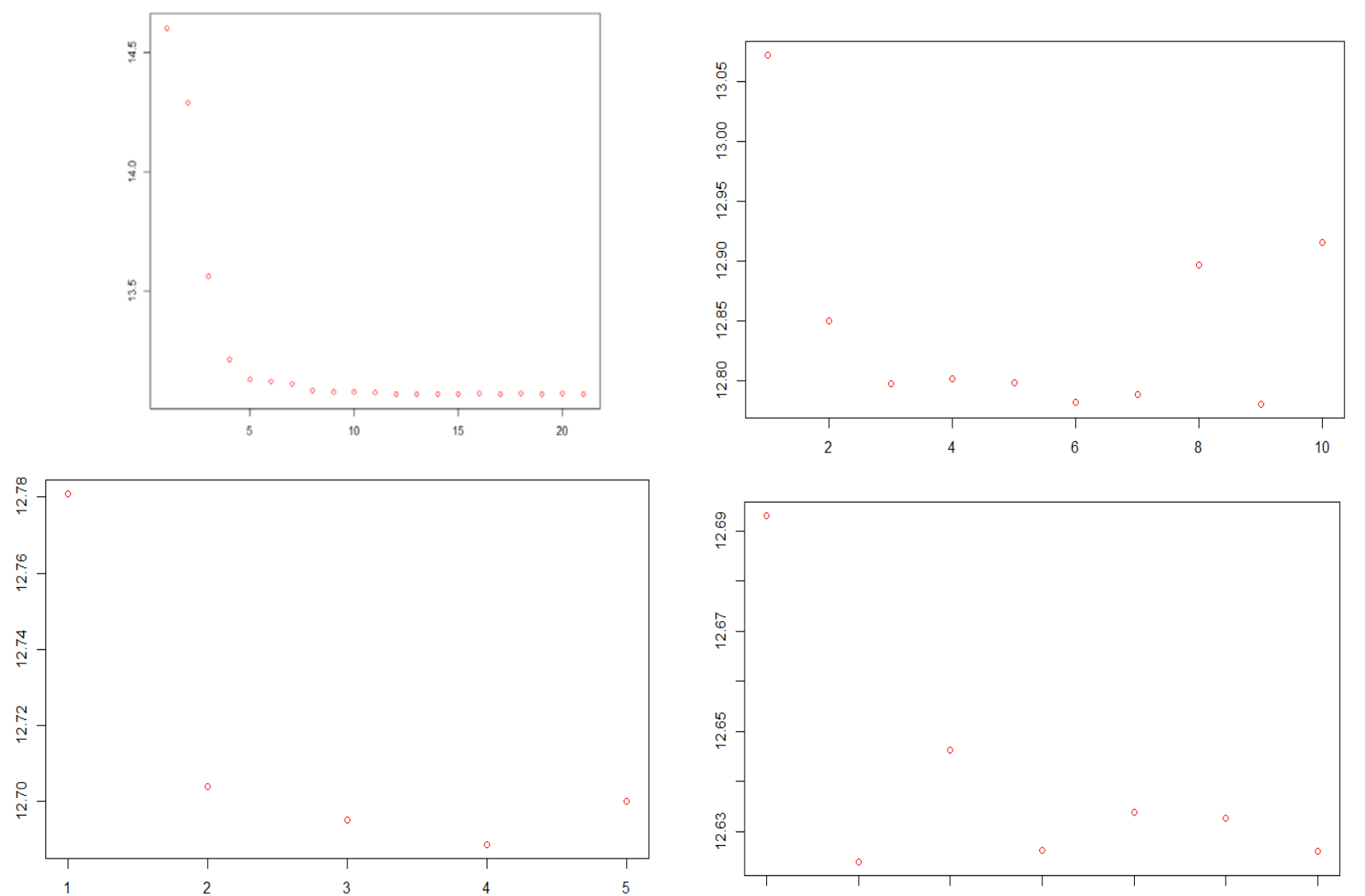
```
Call:
lm(formula = gameDuration ~ towerKills + inhibitorKills + baronKills +
    dragonKills + riftHeraldKills, data = mydata_regression1)

Residuals:
     Min       1Q   Median       3Q      Max
-12.7588  -2.6125  -0.1658   2.3834  22.3490

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      9.97939    0.06039 165.246  < 2e-16 ***
towerKills       1.04208    0.00778 133.938  < 2e-16 ***
inhibitorKills  -0.04978    0.01718  -2.898  0.00376 **
baronKills       1.42911    0.02959  48.290  < 2e-16 ***
dragonKills      3.08956    0.02008 153.834  < 2e-16 ***
riftHeraldKills -1.65113    0.03395 -48.637  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.821 on 51484 degrees of freedom
Multiple R-squared:  0.7996,	Adjusted R-squared:  0.7996
F-statistic: 4.108e+04 on 5 and 51484 DF,  p-value: < 2.2e-16
```

3. Optimize this model using non-linear relationships.

It is possible that these variables has more complicated relationships with the response, so we try polynomial relationship here, and evaluate the new model using the method called "k-fold cross validation".

There are four figures on the right side that indicate the relationship between mean test error and highest power of the polynomial of each variable

When choosing the power with smallest mean test error, we can get a better model:



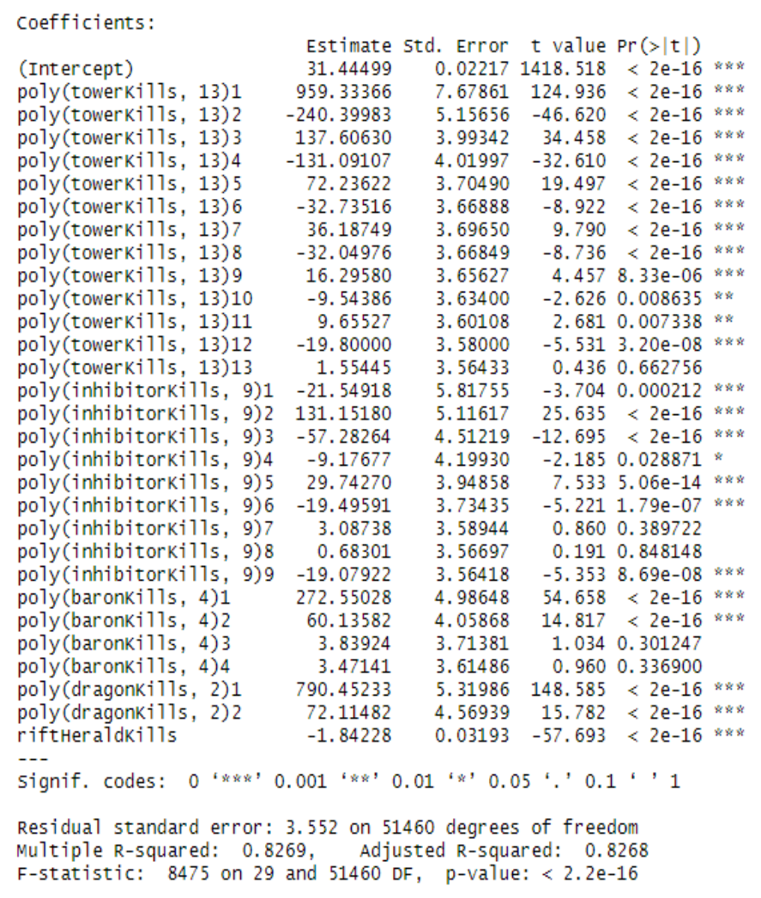
gameDuration ~ poly(towerKills, 13) + poly(inhibitorKills, 9) + poly(baronKills, 4) + poly(dragonKills, 2) + riftHeraldKills

Conclusion and Model Evaluation

1.Conclusion:

We can get quantitative relationships between predictors and the response, here are some of them below:

- With the number of broken towers increasing, the game would probably last longer.
- With the number of baron and dragon killed increasing, the game will also absolutely last longer. This makes sense, because baron and dragon need time to rebirth.
- With the number of Rift Herald killed increasing, the game will last shorter.



```
Coefficients:
                           Estimate Std. Error  t value Pr(>|t|)
(Intercept)                31.44499    0.02217 1418.518  < 2e-16 ***
poly(towerKills, 13)1     959.33366    7.67861  124.936  < 2e-16 ***
poly(towerKills, 13)2    -240.39983    5.15656  -46.620  < 2e-16 ***
poly(towerKills, 13)3     137.60630    3.99342   34.458  < 2e-16 ***
poly(towerKills, 13)4    -131.09107    4.01997  -32.610  < 2e-16 ***
poly(towerKills, 13)5      72.23622    3.70490   19.497  < 2e-16 ***
poly(towerKills, 13)6     -32.73516    3.66888   -8.922  < 2e-16 ***
poly(towerKills, 13)7      36.18749    3.69650    9.790  < 2e-16 ***
poly(towerKills, 13)8     -32.04976    3.66849   -8.736  < 2e-16 ***
poly(towerKills, 13)9      16.29580    3.65627    4.457 8.33e-06 ***
poly(towerKills, 13)10     -9.54386    3.63400   -2.626 0.008635 **
poly(towerKills, 13)11      9.65527    3.60108    2.681 0.007338 **
poly(towerKills, 13)12    -19.80000    3.58000   -5.531 3.20e-08 ***
poly(towerKills, 13)13      1.55445    3.56433    0.436 0.662756
poly(inhibitorKills, 9)1  -21.54918    5.81755   -3.704 0.000212 ***
poly(inhibitorKills, 9)2  131.15180    5.11617   25.635  < 2e-16 ***
poly(inhibitorKills, 9)3  -57.28264    4.51219  -12.695  < 2e-16 ***
poly(inhibitorKills, 9)4   -9.17677    4.19930   -2.185 0.028871 *
poly(inhibitorKills, 9)5   29.74270    3.94858    7.533 5.06e-14 ***
poly(inhibitorKills, 9)6  -19.49591    3.73435   -5.221 1.79e-07 ***
poly(inhibitorKills, 9)7    3.08738    3.58944    0.860 0.389722
poly(inhibitorKills, 9)8    0.68301    3.56697    0.191 0.848148
poly(inhibitorKills, 9)9  -19.07922    3.56418   -5.353 8.69e-08 ***
poly(baronKills, 4)1      272.55028    4.98648   54.658  < 2e-16 ***
poly(baronKills, 4)2       60.13582    4.05868   14.817  < 2e-16 ***
poly(baronKills, 4)3        3.83924    3.71381    1.034 0.301247
poly(baronKills, 4)4        3.47141    3.61486    0.960 0.336900
poly(dragonKills, 2)1     790.45233    5.31986  148.585  < 2e-16 ***
poly(dragonKills, 2)2      72.11482    4.56939   15.782  < 2e-16 ***
riftHeraldKills            -1.84228    0.03193  -57.693  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.552 on 51460 degrees of freedom
Multiple R-squared:  0.8269,	Adjusted R-squared:  0.8268
F-statistic:  8475 on 29 and 51460 DF,  p-value: < 2.2e-16
```

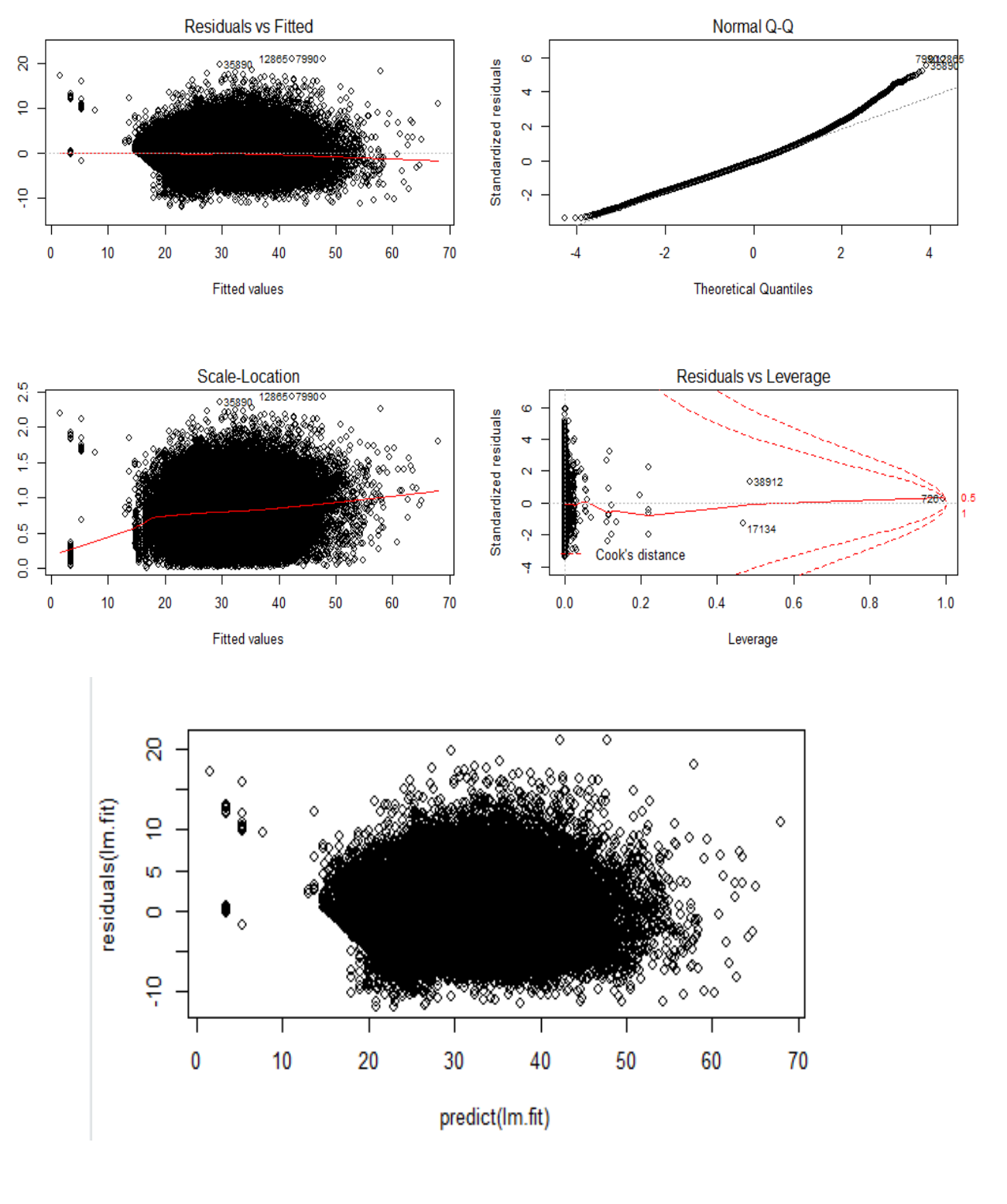
2.Model Evaluation:

- **Accuracy:**

① In this model, adjusted R^2 = 0.8268 which means this model fits the dataset very well;
② RSE = 3.552, which means the error of this model is 3.552 minutes, regarding that the mean of the duration of the games is approximately 31.22minutes, this error seems acceptable.
③ F-statistic = 8475

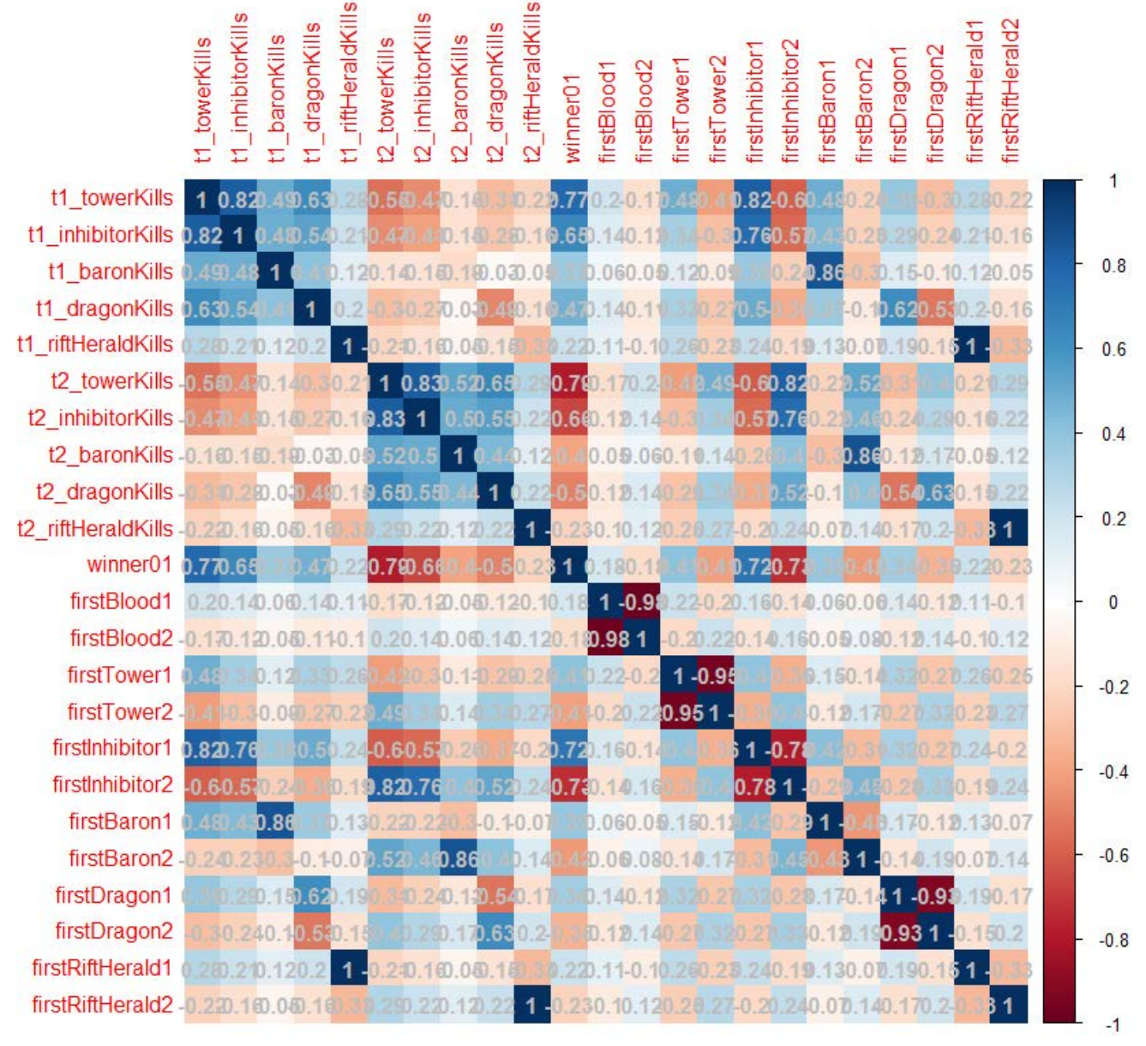
- **Evaluation from the plots:**

① The "Residual vs fitted" indicates that there are some outliers, and the current model fits well.
② The Q-Q plot indicates that the skew of our data is not so strong.
③ In the "Scale Location" plot the read line is pretty flattish, which means data spreading out.
④ The leverage plot suggests that there is an extremely high leverage point in the data.
⑤ The last one indicate that predicting residual is also normal.

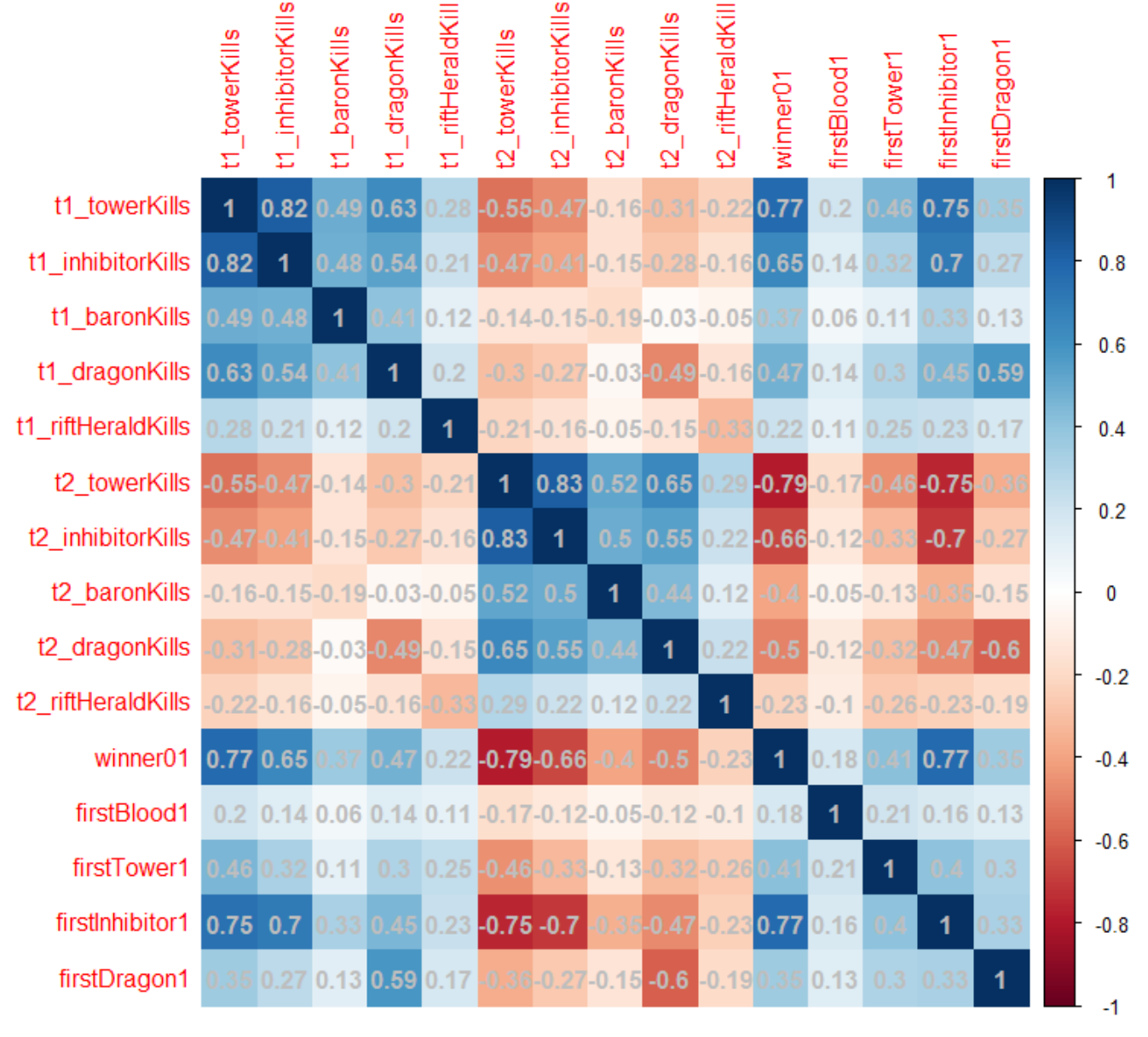


Predicting the Win Rate by Classification

1. Judge Correlation and Remove Factor with High Correlation.



(FIGURE 1)



(FIGURE 2)

- Use *corrplot* function to visualize the correlation and remove 8 predictors with large correlation.
- The Figure 1 is about original correlation and the Figure 2 is the ultimate correlation of left predictors.

2. Classifying by Logistic Regression, LDA and KNN.

	Confusion matrix	Test error

Logistic Regression	0 1
	0 7328 189
	1 370 7560
		3.62%

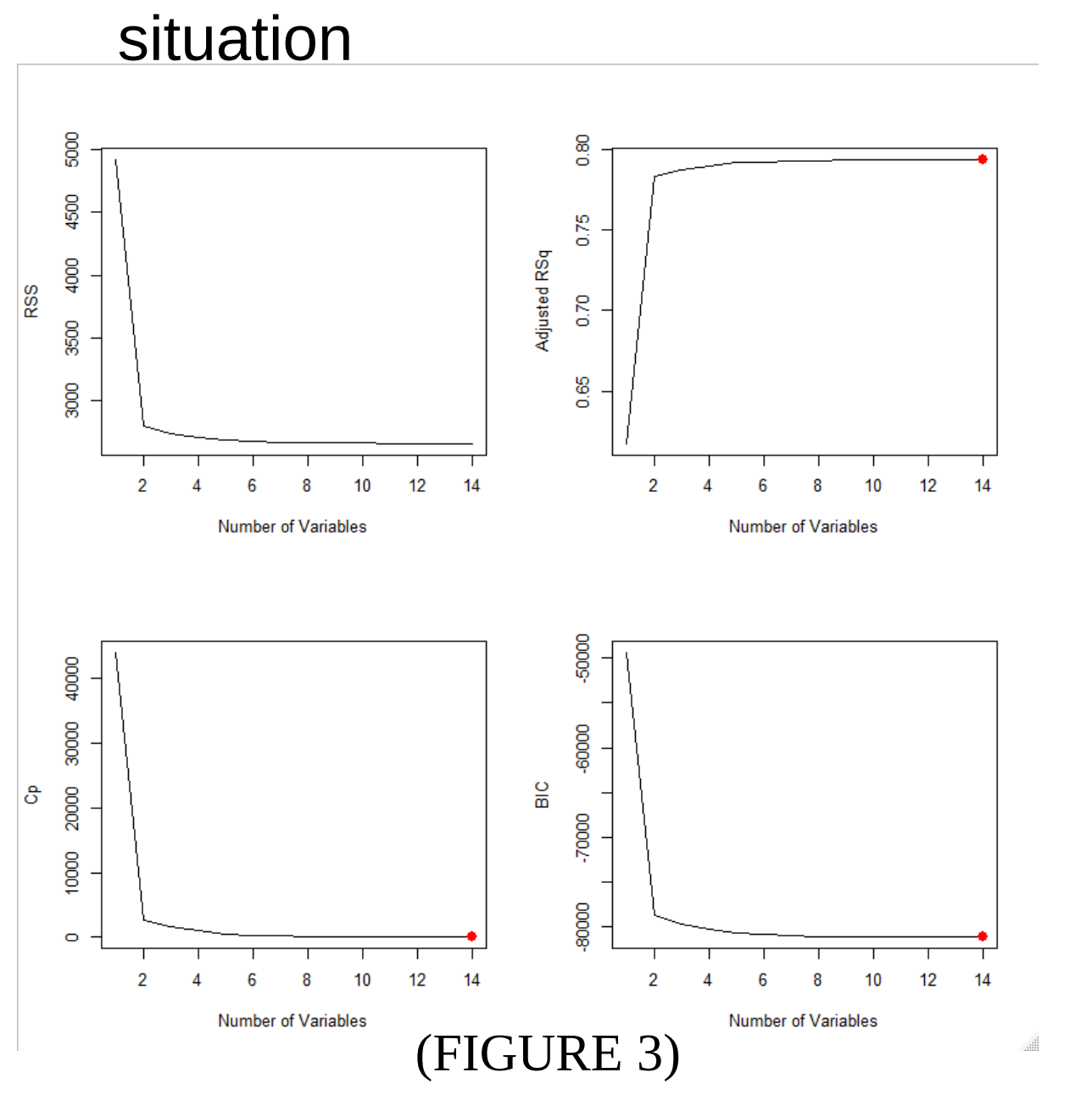
LDA	0 1
	0 7348 245
	1 350 7504
		3.85%

KNN(K = 10)	0 1
	0 7471 321
	1 227 7429
		3.34%

- From the result we can get the conclusion that the KNN model is the best, but their difference is so small. We will optimize the model further at the next step.

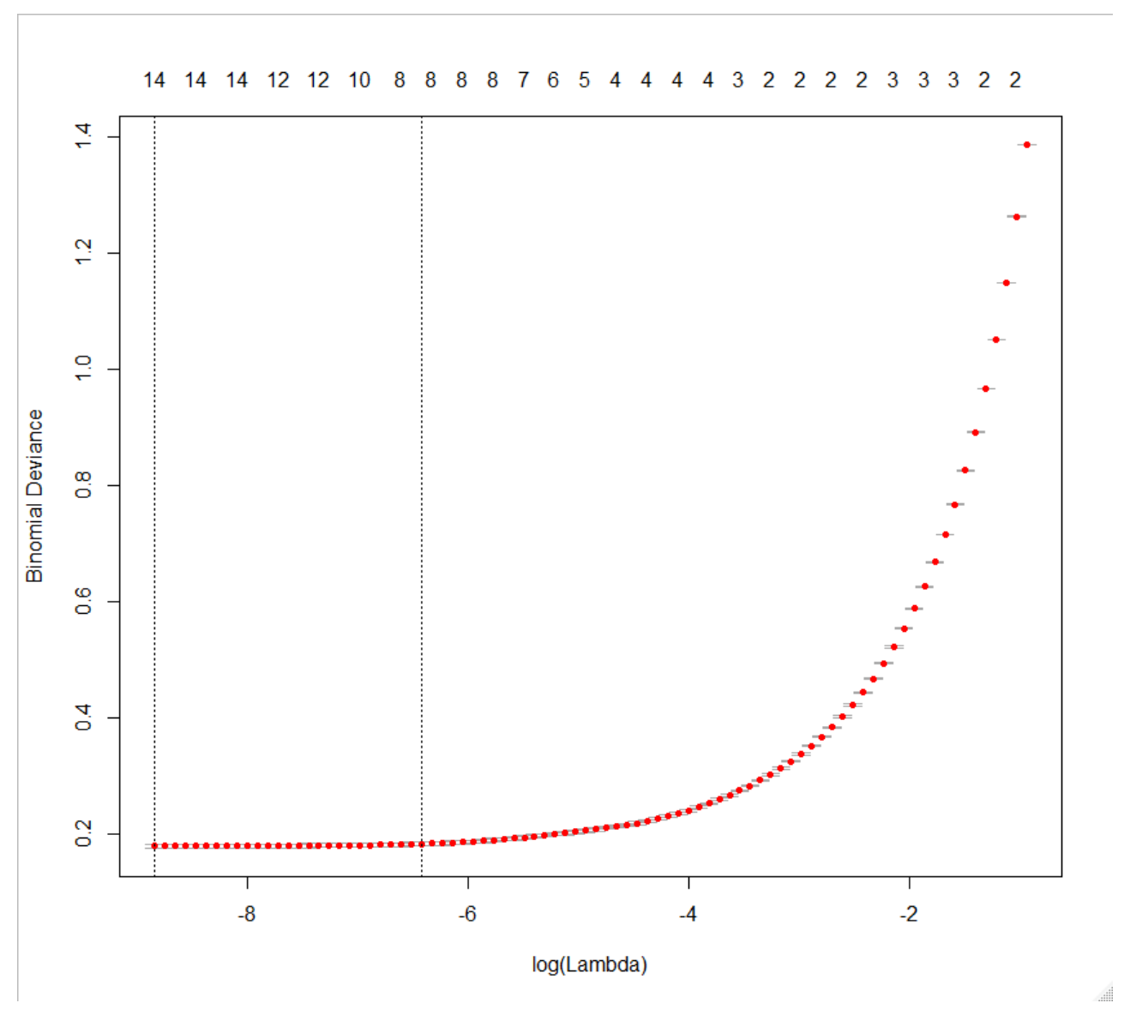
3. Optimize the model by *forward selection* and *the lasso*

- ***Forward Selection:***
  - get the most appopriate number of predictors by the plots.
  - The result is N = 14 is the best situation



(FIGURE 3)
(*R*2, RSS, adjusted *R*2, *Cp*, and BIC change with different number of predictors

- ***The Lasso:* enhance model' interpretability.**
  - When λ = 0.0001441945, the model is best.



(FIGURE 4)
(Binomial deviance versus log(Lambda))

- Under this condition , all coefficients of predictors are 0 except 4 predictors, which make model more interpretable.

4. Analysis of ultimate model

- The final model only correlate to 4 predictors and the test error of logistic regression with these 4 predictors is 4.12% which is a little bigger than original model's , but this model has a stronger interpretation and we can predict the result just by 4 predictors. So the final model is much better.
- The four predictors are the destroyed tower number of team1, the killing baron number of team 1, destroyed tower number of team2, the killing baron number of team 2. Their coefficients are 0.68, 0.2523, -1.623 and -0.7029.