**ORIE 4740 Final Project:**

**League of Legends Game Prediction**

Dong Hyun Kim (dk433), Jiacheng Han (jh995), Yikun Cai (yc2399)

**Abstract**

League of Legends is a multiplayer online arena game, with players assume the role of “summoner” that controls a “champion” and battle against a team of other players. The game is won by destroying the “nexus” of the other team. In 2016, there was an estimate of 100 million active players every month. And the 2017 World Championship had 60 million unique viewers and a total prize pool of over 4 million US dollars. The game is very competitive and we managed to build predictive models to accurately predict the results of a game by looking at significant variables in the game. These variables include amount of deaths, assists, gold earned, and structures (turret) destroyed. To make predictions, we randomly used 2000 data from the League of Legend dataset. The methods we use include Logistic Regression, Forward Subset Selection, Tree-Based Methods (Bagging and Random Forest), and GAM to predict the win or loss for a game. The final model is:

Where:

**Introduction**

Our project examines the dataset of League of Legend ranked matches to determine whether a player will win the match or not. The dataset is taken from Kaggle and includes over one million data points. Each data point will include variables such as player id, win (boolean response variable), kills, killing sprees (number of consecutive kills between deaths), deaths, gold, damage taken and dealt, and structure destruction. The initial data from Kaggle contains 56 variables which all relates to certain aspects of the game.

The proposed data analysis to be performed includes logistic regression analysis, forward subset selection, tree based models, and generalized additive models. Logistic regression will be performed to start the analysis of the model, and significance level of variables and their overall R squared values will be considered. Next, we performed forward subset selection and compared BIC values to decide on the subset of predictors. We also tried a different approach using tree based methods such as bagging and random forest. After determining the final predictors based on importance from random forest, we applied GAM with splines to determine the best model.

In the end, we were able to define a model with 4 predictors (deaths, turretkills, assists, and goldearned) that had a true positive rate of 0.799 and true negative rate of 0.814.

**Dataset Overview**

The dataset League of Legends includes 1 million data entries with 47 variables, including kills, deaths, assists, goldearned, champlvl, etc. We randomly chose 2000 data points to build our models. League is an objective-focused game that accelerates gameplay through buying items with gold. Gold is earned through killing creeps, killing enemy champions, and taking objectives. A few variables that we would be using in the model are explained below:

1. KDA: Kill to death ratio that indicates how well a player is performing relative to enemies and allies. Higher ratio the better.
2. Damage to turrets and turret kills: Indicative of how well a player and their team destroy permanent enemy strongholds. Turrets protect the base; thus, more turret kills means less protection for the enemy base.
3. Wards placed: Vision is the most important aspect in League, so more wards means more information for your team to act on.

**Models and Results**

1. Logistic Regression

Before we started modeling, we cleaned up the dataset. From the dataset from Kaggle, there were several predictors that did not pertain to our model or had null entries. Furthermore, we needed to ensure that the response variable, “win”, was a categorical factor and not a numerical value of 0 or 1. We used simple logistic regression to check on the significant levels of each variables except "id", "item 1","item 2","item 3","item 4","item 5","item 6","trinket", because these columns have categorical values that are unrelated to the prediction. After we run the logistic regression, these variables are significant with their respective p-values smaller than 0.05:

Kills, Deaths, Assists, Logesttimespentliving, Quadrakills, Totheal, Totunitshealed, Dmgtoturrets, Goldearned, Turretkills, Inhibkills, Totminionskilled, Neutralminionskilled, Ownjunglekills, Wardsplaced.

For our predictions, we set the threshold to be 0.5. Table 1.1 shows the confusion matrix for our initial model. True positive rate is 0.755 and true negative rate is 0.711.

|  |  |  |
| --- | --- | --- |
| Initial Model  - Logistic Regression | 0 | 1 |
| 0 | 711 | 245 |
| 1 | 289 | 755 |

Table 1.1: Confusion matrix for initial log regression model

After we narrowed the number of relevant predictors from 56 to 15 mentioned above, we decided to run a forward subset selection to narrow it down even further. We compared two criterions for determining the number of predictors: Adjusted R Squared and BIC. Figure 1.2 shows the comparison between these two criterions. Adjusted R Squared resulted in selecting 15 predictors while BIC resulted in selecting 12 predictors. In the end, we decided to the BIC criterion because it penalizes higher predictor models more heavily than Adjusted R Squared. The 12 predictors selected using forward selection (BIC) are:

kills, deaths, assists, longgesttimespentliving, totunitsheals, goldearned, turretkills, inhibkills, totminionskilled, neutralminionskilled, ownjunglekills, and wardsplaced

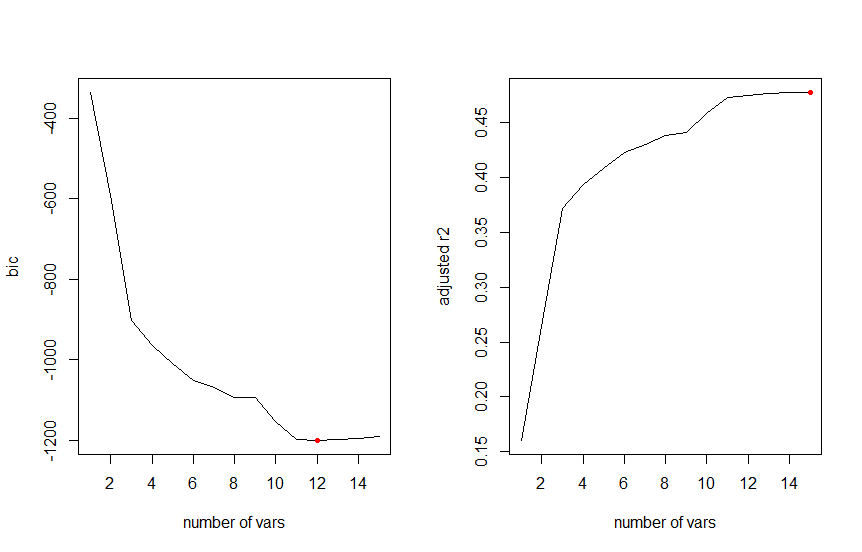


Figure 1.2: Forward Subset Selection based on BIC and Adjusted R Squared

Table 1.3 shows the confusion matrix for the 12 predictor model. The true positive rate is 0.786 and true negative rate is 0.718. There was no significant improvement from the initial logistic regression model.

|  |  |  |
| --- | --- | --- |
| 12 predictor - forward subset selection | 0 | 1 |
| 0 | 718 | 214 |
| 1 | 282 | 786 |

Table 1.3: Confusion matrix for 12 predictor model based on forward subset selection

Tree-Based Methods

Because we didn’t see significant improvement using logistic regression and forward subset selection, we decided to use tree based methods. With random sampling, we grow a tree based classification model as shown in Figure 2.1.

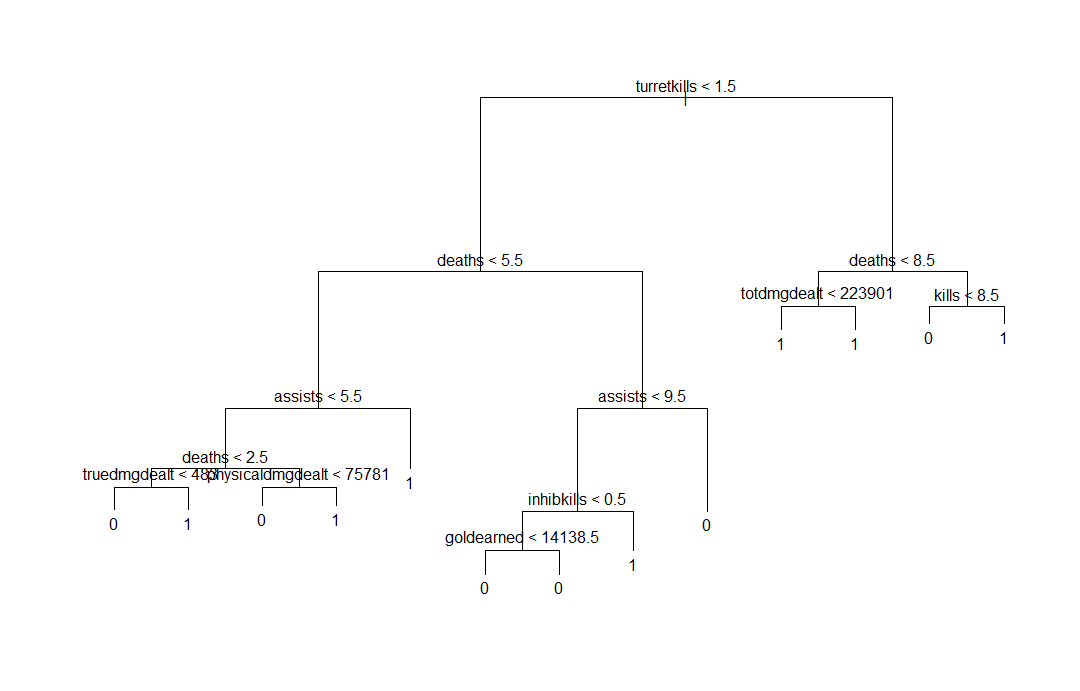


Figure 2.1: Dendrogram of grown tree with random sampling

To determine which subtree to prune the tree to, we performed cross validation analysis to find the tree with the lowest variance. Figure 2.2 shows the cross validation analysis. We can see that subtree with size 10 yields the lowest variance. Figure 2.3 shows the tree after being pruned to 10 nodes.

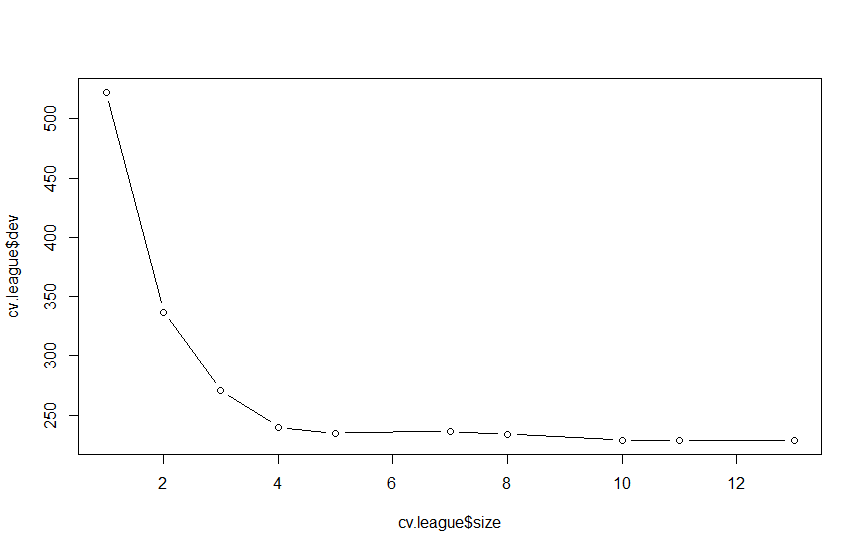


Figure 2.2: Cross validation analysis for pruning

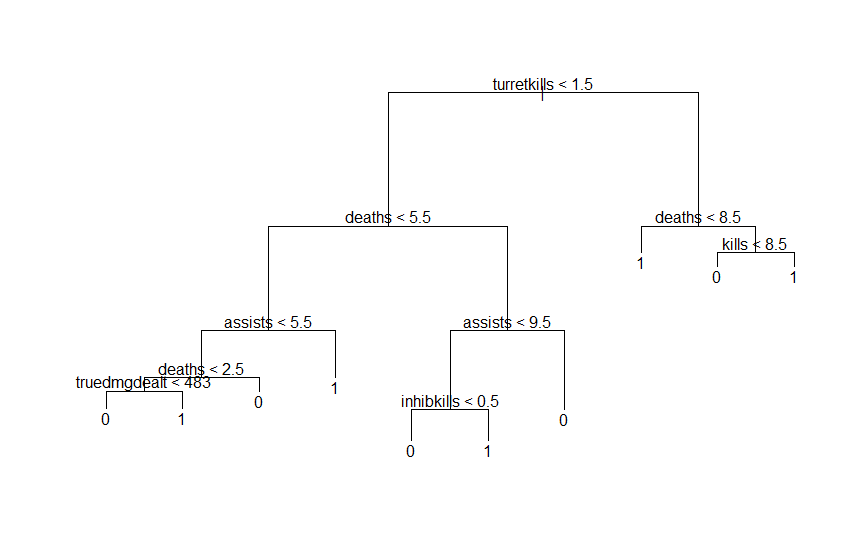


Figure 2.3: Dendrogram for pruned tree

> summary(tree.league)

Classification tree:

tree(formula = win ~ ., data = league, subset = train)

Variables actually used in tree construction:

[1] "turretkills" "deaths" "assists" "truedmgdealt" "physicaldmgdealt" "inhibkills" "goldearned" "totdmgdealt" "kills"

Number of terminal nodes: 13

Residual mean deviance: 0.8147 = 804.1 / 987

Misclassification error rate: 0.199 = 199 / 1000

> summary(prune.league)

Classification tree:

snip.tree(tree = tree.league, nodes = c(6L, 20L, 17L))

Variables actually used in tree construction:

[1] "turretkills" "deaths" "assists" "truedmgdealt" "inhibkills" "kills"

Number of terminal nodes: 10

Residual mean deviance: 0.8684 = 859.7 / 990

Misclassification error rate: 0.2 = 200 / 1000

From the above console output, we can see that pruning the tree resulting in the residual mean deviance increasing from 0.8147 to 0.8684. The misclassification error rate increased very slightly from 0.199 to 0.2, which is negligible difference. Then, we decided to perform random forest method, which will reduce the variance further and may result in a different set of predictors.

1. Random Forest

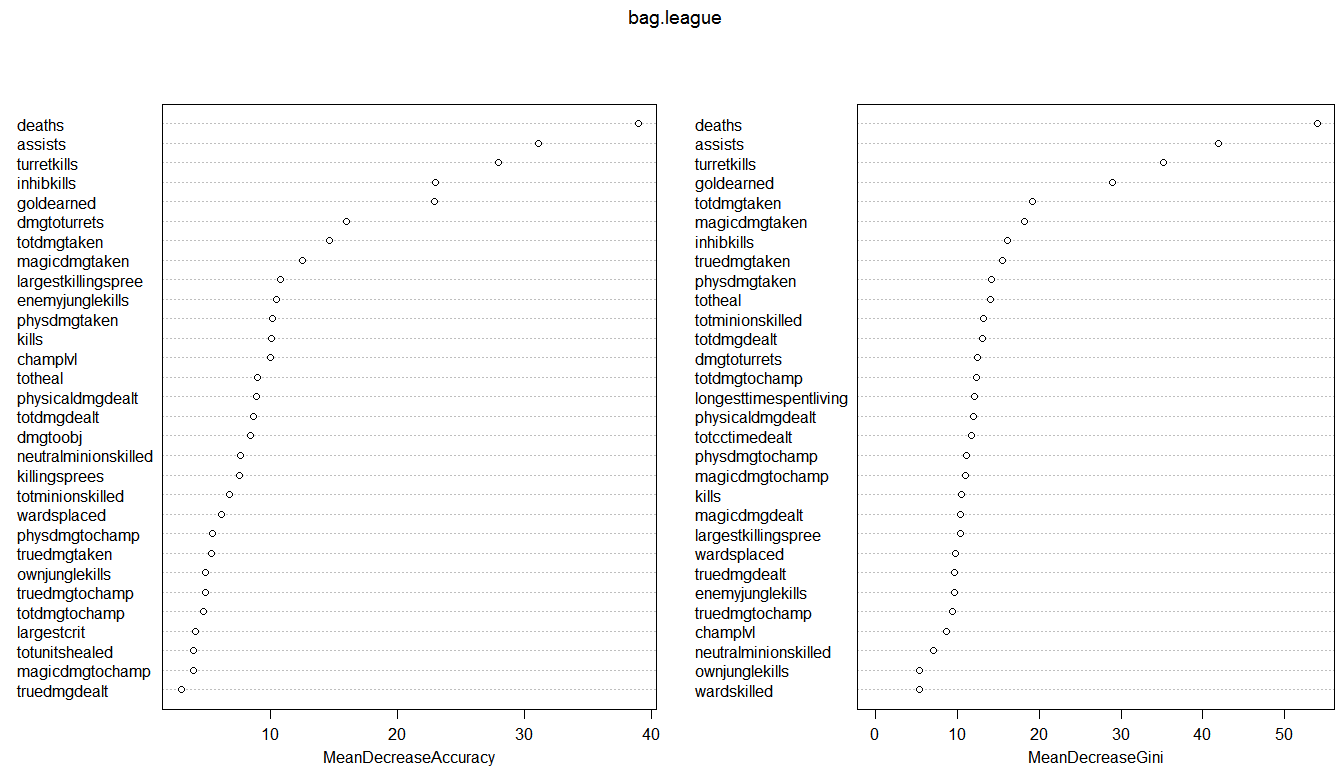
In general, a single decision tree has high variance and could potentially encounter overfitting problem as well. Then we built random forest model. Figure 3.1 shows the importance of the predictors in the random forest model. Using the MeanDecreaseGini criterion, the most far apart variables (deaths, turretkills, assists, goldearned) are chosen because they explained the most variance of the data.

Figure 3.1 Accuracy and Gini importance plot

1. GAM Model

GAM model was utilized to include the 4 variables mentioned above (deaths, turretkills, assists, goldearned). In reference to table 3.1, first, we built gam model to fit deaths using a smoothing spline with degrees of freedom of 2, 4, and 6, comparing to a model that does not have the smoothing spline. Other three functions are fitted using a different values for each dummy variables associated with turretkills, assists, goldearned respectively. By using anova test, we found out that the p-value of second model is <0.001, there's a significant difference between the second and first model, which means including the deaths as smoothing spline with 2 degrees of freedom is making a difference. When comparing the third model and second model, there's again significant difference between these two models, because the p-value is <0.001. When comparing the fourth model and third model, there's no significant difference between these two models, because the p-value is >0.05. So we picked the model with smoothing spline on deaths with 4 degrees of freedom.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Variable 1:  deaths | Variable 2:  turretkills | Variable 3:  assists | Variable 4:  goldearned | Whether P-value is significant |
| Model 1 | deaths | turretkills | assists | goldearned |  |
| Model 2 | Smoothing Spline (deaths) DOF 2 | turretkills | assists | goldearned | Significant P-value |
| Model 3 | Smoothing Spline (deaths) DOF 4 | turretkills | assists | goldearned | Significant P-value |
| Model 4 | Smoothing Spline (deaths) DOF 6 | turretkills | assists | goldearned | Insignificant P-value |

Table 3.1 GAM Model Selection on Deaths

Similarly, we repeated same procedure with turretkills, assists, goldearned. The results are shown in table 3.2 below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Variable 1:  deaths | Variable 2:  turretkills | Variable 3:  assists | Variable 4:  goldearned | Whether P-value is significant |
| Model 1 | Smoothing Spline (deaths) DOF 4 | turretkills | assists | goldearned |  |
| Model 2 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | assists | goldearned | Significant P-value |
| Model 3 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 4 | assists | goldearned | Insignificant P-value |
| Model 4 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 6 | assists | goldearned | Insignificant P-value |
| Model 5 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | assists | goldearned |  |
| Model 6 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | Smoothing Spline (assists) DOF 2 | goldearned | Significant P-value |
| Model 7 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | Smoothing Spline (assists) DOF 4 | goldearned | Insignificant P-value |
| Model 8 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | Smoothing Spline (assists) DOF 6 | goldearned | Insignificant P-value |
| Model 9 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | Smoothing Spline (assists) DOF 2 | goldearned |  |
| Model 10 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | Smoothing Spline (assists) DOF 2 | Smoothing Spline (goldearned) DOF 2 | Significant P-value |
| Model 11 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | Smoothing Spline (assists) DOF 2 | Smoothing Spline (goldearned) DOF 4 | Insignificant P-value |
| Model 12 | Smoothing Spline (deaths) DOF 4 | Smoothing Spline (turretkills) DOF 2 | Smoothing Spline (assists) DOF 2 | Smoothing Spline (goldearned) DOF 6 | Insignificant P-value |

Table 3.2 GAM Model Selection on turretkills, assists, goldearned

The final model includes smoothing splines on deaths with 4 degree of freedom, smoothing splines on turretkills with 2 degree of freedom, smoothing splines on assists with 2 degree of freedom, and smoothing splines on goldearned with 2 degree of freedom. The true positive rate and true negative rate are shown in the Table 3.3, which are 0.799 and 0.814 respectively. In comparison to the logistic model derived from forward subset selection, which had TPR = 0.786 and TNR = 0.718, the TPR and TNR of the GAM smoothing splines model is much better.

|  |  |  |
| --- | --- | --- |
| Final Model: GAM | 0 | 1 |
| 0 | 814 | 201 |
| 1 | 186 | 799 |

Table 3.3: Confusion matrix for final model with smoothing splines from GAM

**Conclusion**

We were aware of some correlations based on our pre-knowledge of the game. With the preliminary logistic regression technique, the 12 variables chosen made intuitive sense in how they affected win probability. However, without cross-validation, it was inconclusive and not as accurate as other methods. The decision tree helped to visualize the win probability based on a complexity parameter of 4. We see that high turret kills and low deaths were indicative of winning.

Ultimately, random forest was the best base model to go with. We set parameter “mtry” to 7, which is close to the square root of 56, to differentiate from the bagging method. 4 variables were chosen as explaining majority of the variance in the data. From there we added interaction terms through GAMs. We decided to use smooth splines with degrees and freedom ranging from 2 to 6 on each of the 4 variables. We compared the 4 models (no spline, spline with degrees of freedom 2, 4, and 6) using ANOVA tests and compared p-values. Repeating this step 4 times for each variables, we finalized a model that had more correct predictions than the other methods.

It was interesting to see the progression of the accuracy of the models and conclude on a model with around 80% accuracy.

**References**

Textbook: An Intro to Statistical Learning with Applications in R

Dataset: <https://www.kaggle.com/paololol/league-of-legends-ranked-matches>

R Packages: tree, MASS, randomforest, gam, splines