ML Project

Luis Gjuraj & Lo-Badal Burch

4/11/2022

# Let’s do some R coding.

Let’s load in the data first. Change the pathing to your data.

lol\_data <- read.csv("data ml/games.csv")  
lol\_data$t2\_win <- lol\_data$winner-1  
lol\_champ <- read.csv("data ml/champion2\_better.csv")  
lol\_spell <- read.csv("data ml/summoner\_spell.csv")

Let’s turn JSON files into dataframes. We did that in python.(R not good…)

We can see that we have 3 datasets ready for use. The first one, which we called lol\_data, has a lot of useful information about how the game progresses, and important events that either team 1 or team 2 won. We will further use this data using different models to determine the win or loss of a game based on these parameters.

We also have lol\_champ data that gives us insight on each champion, by pairing their id to their names, and lol\_spell dataset that does the same thing for summoner spells used.

Each champion has different traits and skills, and each spell has different uses and effects.

head(lol\_data)

## gameId creationTime gameDuration seasonId winner firstBlood firstTower  
## 1 3326086514 1.504279e+12 1949 9 1 2 1  
## 2 3229566029 1.497849e+12 1851 9 1 1 1  
## 3 3327363504 1.504360e+12 1493 9 1 2 1  
## 4 3326856598 1.504349e+12 1758 9 1 1 1  
## 5 3330080762 1.504554e+12 2094 9 1 2 1  
## 6 3287435705 1.501668e+12 2059 9 1 2 2  
## firstInhibitor firstBaron firstDragon firstRiftHerald t1\_champ1id  
## 1 1 1 1 2 8  
## 2 1 0 1 1 119  
## 3 1 1 2 0 18  
## 4 1 1 1 0 57  
## 5 1 1 1 0 19  
## 6 1 1 2 0 40  
## t1\_champ1\_sum1 t1\_champ1\_sum2 t1\_champ2id t1\_champ2\_sum1 t1\_champ2\_sum2  
## 1 12 4 432 3 4  
## 2 7 4 39 12 4  
## 3 4 7 141 11 4  
## 4 4 12 63 4 14  
## 5 4 12 29 11 4  
## 6 3 4 141 11 4  
## t1\_champ3id t1\_champ3\_sum1 t1\_champ3\_sum2 t1\_champ4id t1\_champ4\_sum1  
## 1 96 4 7 11 11  
## 2 76 4 3 10 4  
## 3 267 3 4 68 4  
## 4 29 4 7 61 4  
## 5 40 4 3 119 4  
## 6 24 12 4 45 3  
## t1\_champ4\_sum2 t1\_champ5id t1\_champ5\_sum1 t1\_champ5\_sum2 t1\_towerKills  
## 1 6 112 4 14 11  
## 2 14 35 4 11 10  
## 3 12 38 12 4 8  
## 4 1 36 11 4 9  
## 5 7 134 7 4 9  
## 6 4 67 4 7 8  
## t1\_inhibitorKills t1\_baronKills t1\_dragonKills t1\_riftHeraldKills t1\_ban1  
## 1 1 2 3 0 92  
## 2 4 0 2 1 51  
## 3 1 1 1 0 117  
## 4 2 1 2 0 238  
## 5 2 1 3 0 90  
## 6 1 1 1 0 117  
## t1\_ban2 t1\_ban3 t1\_ban4 t1\_ban5 t2\_champ1id t2\_champ1\_sum1 t2\_champ1\_sum2  
## 1 40 69 119 141 104 11 4  
## 2 122 17 498 19 54 4 12  
## 3 40 29 16 53 69 4 7  
## 4 67 516 114 31 90 14 4  
## 5 64 412 25 31 37 3 4  
## 6 6 238 122 105 92 4 12  
## t2\_champ2id t2\_champ2\_sum1 t2\_champ2\_sum2 t2\_champ3id t2\_champ3\_sum1  
## 1 498 4 7 122 6  
## 2 25 4 14 120 11  
## 3 412 14 4 126 4  
## 4 19 11 4 412 4  
## 5 59 4 12 141 11  
## 6 15 4 7 245 12  
## t2\_champ3\_sum2 t2\_champ4id t2\_champ4\_sum1 t2\_champ4\_sum2 t2\_champ5id  
## 1 4 238 14 4 412  
## 2 4 157 4 14 92  
## 3 12 24 4 11 22  
## 4 3 92 4 14 22  
## 5 4 38 4 12 51  
## 6 4 2 4 11 12  
## t2\_champ5\_sum1 t2\_champ5\_sum2 t2\_towerKills t2\_inhibitorKills t2\_baronKills  
## 1 4 3 5 0 0  
## 2 4 7 2 0 0  
## 3 7 4 2 0 0  
## 4 4 7 0 0 0  
## 5 4 7 3 0 0  
## 6 4 14 6 0 0  
## t2\_dragonKills t2\_riftHeraldKills t2\_ban1 t2\_ban2 t2\_ban3 t2\_ban4 t2\_ban5  
## 1 1 1 114 67 43 16 51  
## 2 0 0 11 67 238 51 420  
## 3 1 0 157 238 121 57 28  
## 4 0 0 164 18 141 40 51  
## 5 1 0 86 11 201 122 18  
## 6 3 0 119 134 154 63 31  
## t2\_win  
## 1 0  
## 2 0  
## 3 0  
## 4 0  
## 5 0  
## 6 0

names(lol\_data)

## [1] "gameId" "creationTime" "gameDuration"   
## [4] "seasonId" "winner" "firstBlood"   
## [7] "firstTower" "firstInhibitor" "firstBaron"   
## [10] "firstDragon" "firstRiftHerald" "t1\_champ1id"   
## [13] "t1\_champ1\_sum1" "t1\_champ1\_sum2" "t1\_champ2id"   
## [16] "t1\_champ2\_sum1" "t1\_champ2\_sum2" "t1\_champ3id"   
## [19] "t1\_champ3\_sum1" "t1\_champ3\_sum2" "t1\_champ4id"   
## [22] "t1\_champ4\_sum1" "t1\_champ4\_sum2" "t1\_champ5id"   
## [25] "t1\_champ5\_sum1" "t1\_champ5\_sum2" "t1\_towerKills"   
## [28] "t1\_inhibitorKills" "t1\_baronKills" "t1\_dragonKills"   
## [31] "t1\_riftHeraldKills" "t1\_ban1" "t1\_ban2"   
## [34] "t1\_ban3" "t1\_ban4" "t1\_ban5"   
## [37] "t2\_champ1id" "t2\_champ1\_sum1" "t2\_champ1\_sum2"   
## [40] "t2\_champ2id" "t2\_champ2\_sum1" "t2\_champ2\_sum2"   
## [43] "t2\_champ3id" "t2\_champ3\_sum1" "t2\_champ3\_sum2"   
## [46] "t2\_champ4id" "t2\_champ4\_sum1" "t2\_champ4\_sum2"   
## [49] "t2\_champ5id" "t2\_champ5\_sum1" "t2\_champ5\_sum2"   
## [52] "t2\_towerKills" "t2\_inhibitorKills" "t2\_baronKills"   
## [55] "t2\_dragonKills" "t2\_riftHeraldKills" "t2\_ban1"   
## [58] "t2\_ban2" "t2\_ban3" "t2\_ban4"   
## [61] "t2\_ban5" "t2\_win"

head(lol\_champ)

## tags title id key  
## 1 ['Fighter', 'Tank'] the Monkey King 62 MonkeyKing  
## 2 ['Fighter', 'Assassin'] Grandmaster at Arms 24 Jax  
## 3 ['Mage', 'Support'] the Harbinger of Doom 9 Fiddlesticks  
## 4 ['Assassin'] the Demon Jester 35 Shaco  
## 5 ['Fighter', 'Tank'] the Uncaged Wrath of Zaun 19 Warwick  
## 6 ['Marksman'] the Rebel 498 Xayah  
## name  
## 1 Wukong  
## 2 Jax  
## 3 Fiddlesticks  
## 4 Shaco  
## 5 Warwick  
## 6 Xayah

head(lol\_spell)

## id summonerLevel name key  
## 1 1 6 Cleanse SummonerBoost  
## 2 3 4 Exhaust SummonerExhaust  
## 3 4 8 Flash SummonerFlash  
## 4 6 1 Ghost SummonerHaste  
## 5 7 1 Heal SummonerHeal  
## 6 11 10 Smite SummonerSmite  
## description  
## 1 Removes all disables (excluding suppression and airborne) and summoner spell debuffs affecting your champion and lowers the duration of incoming disables by 65% for 3 seconds.  
## 2 Exhausts target enemy champion, reducing their Movement Speed by 30%, and their damage dealt by 40% for 2.5 seconds.  
## 3 Teleports your champion a short distance toward your cursor's location.  
## 4 Your champion gains increased Movement Speed and can move through units for 10 seconds. Grants a maximum of 28-45% (depending on champion level) Movement Speed after accelerating for 2 seconds.  
## 5 Restores 90-345 Health (depending on champion level) and grants 30% Movement Speed for 1 second to you and target allied champion. This healing is halved for units recently affected by Summoner Heal.  
## 6 Deals 390-1000 true damage (depending on champion level) to target epic, large, or medium monster or enemy minion. Restores Health based on your maximum life when used against monsters.

Looks good, but we don’t need a game ID or season. Let’s take those out.

We can see that in the lol\_data dataset, we have ban choices for every player. What those choices basically mean, is that before every game, each player gets to choose a champion that they don’t want to play against. That is a crucial part of the game, but a lot of players don’t know about champions that will put them at a disadvantage. So, using KNN, we will help them figure out which champion to ban at the start!

lol\_data <- lol\_data[,-c(1,2,4)]  
head(lol\_data)

## gameDuration winner firstBlood firstTower firstInhibitor firstBaron  
## 1 1949 1 2 1 1 1  
## 2 1851 1 1 1 1 0  
## 3 1493 1 2 1 1 1  
## 4 1758 1 1 1 1 1  
## 5 2094 1 2 1 1 1  
## 6 2059 1 2 2 1 1  
## firstDragon firstRiftHerald t1\_champ1id t1\_champ1\_sum1 t1\_champ1\_sum2  
## 1 1 2 8 12 4  
## 2 1 1 119 7 4  
## 3 2 0 18 4 7  
## 4 1 0 57 4 12  
## 5 1 0 19 4 12  
## 6 2 0 40 3 4  
## t1\_champ2id t1\_champ2\_sum1 t1\_champ2\_sum2 t1\_champ3id t1\_champ3\_sum1  
## 1 432 3 4 96 4  
## 2 39 12 4 76 4  
## 3 141 11 4 267 3  
## 4 63 4 14 29 4  
## 5 29 11 4 40 4  
## 6 141 11 4 24 12  
## t1\_champ3\_sum2 t1\_champ4id t1\_champ4\_sum1 t1\_champ4\_sum2 t1\_champ5id  
## 1 7 11 11 6 112  
## 2 3 10 4 14 35  
## 3 4 68 4 12 38  
## 4 7 61 4 1 36  
## 5 3 119 4 7 134  
## 6 4 45 3 4 67  
## t1\_champ5\_sum1 t1\_champ5\_sum2 t1\_towerKills t1\_inhibitorKills t1\_baronKills  
## 1 4 14 11 1 2  
## 2 4 11 10 4 0  
## 3 12 4 8 1 1  
## 4 11 4 9 2 1  
## 5 7 4 9 2 1  
## 6 4 7 8 1 1  
## t1\_dragonKills t1\_riftHeraldKills t1\_ban1 t1\_ban2 t1\_ban3 t1\_ban4 t1\_ban5  
## 1 3 0 92 40 69 119 141  
## 2 2 1 51 122 17 498 19  
## 3 1 0 117 40 29 16 53  
## 4 2 0 238 67 516 114 31  
## 5 3 0 90 64 412 25 31  
## 6 1 0 117 6 238 122 105  
## t2\_champ1id t2\_champ1\_sum1 t2\_champ1\_sum2 t2\_champ2id t2\_champ2\_sum1  
## 1 104 11 4 498 4  
## 2 54 4 12 25 4  
## 3 69 4 7 412 14  
## 4 90 14 4 19 11  
## 5 37 3 4 59 4  
## 6 92 4 12 15 4  
## t2\_champ2\_sum2 t2\_champ3id t2\_champ3\_sum1 t2\_champ3\_sum2 t2\_champ4id  
## 1 7 122 6 4 238  
## 2 14 120 11 4 157  
## 3 4 126 4 12 24  
## 4 4 412 4 3 92  
## 5 12 141 11 4 38  
## 6 7 245 12 4 2  
## t2\_champ4\_sum1 t2\_champ4\_sum2 t2\_champ5id t2\_champ5\_sum1 t2\_champ5\_sum2  
## 1 14 4 412 4 3  
## 2 4 14 92 4 7  
## 3 4 11 22 7 4  
## 4 4 14 22 4 7  
## 5 4 12 51 4 7  
## 6 4 11 12 4 14  
## t2\_towerKills t2\_inhibitorKills t2\_baronKills t2\_dragonKills  
## 1 5 0 0 1  
## 2 2 0 0 0  
## 3 2 0 0 1  
## 4 0 0 0 0  
## 5 3 0 0 1  
## 6 6 0 0 3  
## t2\_riftHeraldKills t2\_ban1 t2\_ban2 t2\_ban3 t2\_ban4 t2\_ban5 t2\_win  
## 1 1 114 67 43 16 51 0  
## 2 0 11 67 238 51 420 0  
## 3 0 157 238 121 57 28 0  
## 4 0 164 18 141 40 51 0  
## 5 0 86 11 201 122 18 0  
## 6 0 119 134 154 63 31 0

colnames(lol\_data)

## [1] "gameDuration" "winner" "firstBlood"   
## [4] "firstTower" "firstInhibitor" "firstBaron"   
## [7] "firstDragon" "firstRiftHerald" "t1\_champ1id"   
## [10] "t1\_champ1\_sum1" "t1\_champ1\_sum2" "t1\_champ2id"   
## [13] "t1\_champ2\_sum1" "t1\_champ2\_sum2" "t1\_champ3id"   
## [16] "t1\_champ3\_sum1" "t1\_champ3\_sum2" "t1\_champ4id"   
## [19] "t1\_champ4\_sum1" "t1\_champ4\_sum2" "t1\_champ5id"   
## [22] "t1\_champ5\_sum1" "t1\_champ5\_sum2" "t1\_towerKills"   
## [25] "t1\_inhibitorKills" "t1\_baronKills" "t1\_dragonKills"   
## [28] "t1\_riftHeraldKills" "t1\_ban1" "t1\_ban2"   
## [31] "t1\_ban3" "t1\_ban4" "t1\_ban5"   
## [34] "t2\_champ1id" "t2\_champ1\_sum1" "t2\_champ1\_sum2"   
## [37] "t2\_champ2id" "t2\_champ2\_sum1" "t2\_champ2\_sum2"   
## [40] "t2\_champ3id" "t2\_champ3\_sum1" "t2\_champ3\_sum2"   
## [43] "t2\_champ4id" "t2\_champ4\_sum1" "t2\_champ4\_sum2"   
## [46] "t2\_champ5id" "t2\_champ5\_sum1" "t2\_champ5\_sum2"   
## [49] "t2\_towerKills" "t2\_inhibitorKills" "t2\_baronKills"   
## [52] "t2\_dragonKills" "t2\_riftHeraldKills" "t2\_ban1"   
## [55] "t2\_ban2" "t2\_ban3" "t2\_ban4"   
## [58] "t2\_ban5" "t2\_win"

lol\_ban <- lol\_data[,c(29:34,54:58)]  
head(lol\_ban)

## t1\_ban1 t1\_ban2 t1\_ban3 t1\_ban4 t1\_ban5 t2\_champ1id t2\_ban1 t2\_ban2 t2\_ban3  
## 1 92 40 69 119 141 104 114 67 43  
## 2 51 122 17 498 19 54 11 67 238  
## 3 117 40 29 16 53 69 157 238 121  
## 4 238 67 516 114 31 90 164 18 141  
## 5 90 64 412 25 31 37 86 11 201  
## 6 117 6 238 122 105 92 119 134 154  
## t2\_ban4 t2\_ban5  
## 1 16 51  
## 2 51 420  
## 3 57 28  
## 4 40 51  
## 5 122 18  
## 6 63 31

1. KNN

At first we tried separating champion ids and ban picks using r, but it wasn’t easy to append them to each other in r. The reason why we want to do that is because we want a dataframe of 2 columns. One will be the ids of the champion the player is using, and the other will be the id of the champion the player wants to ban. Having a dataset with 2 sets of variables makes the use of KNN possible for us in this case.

Now that we have usable datasets, let us show off our data analysis skills!

First off, let’s get suggestions on what champion to ban based on the champion we want to play as.

We do not have any data about our champions’ abilities and statistics to find out which other champions would be a bad match to face off against, so we will simply take someone elses opinion. By that, I mean everyone else who plays the same champion that we are. We will be using K-nearest neighbors for this one!

The first step will be using our costumized dataset for our KNN implementation. Check out the python script to see how we made it!

We then split the data into training (80% of the whole dataset )and test data( remaining 20%).

We then created a model using the training data, and we tested it out on our test data. The results were pretty good.

# We are using Champion ID for our X axis and their Ban picks for our Y axis,   
# because that is what we want to predict.  
  
KNN\_data <- read.csv("data ml/KNN\_data.csv")  
#KNN\_data  
v <- sort(sample(1:nrow(KNN\_data),.8\*nrow(KNN\_data)))  
knntrain <- KNN\_data[v,]  
knntest <- KNN\_data[-v,]

We wanted to try different k values, and this is how we did it.

With the KNN models up and running, we test them out and find the best!

#KNN.pred <- knn(knntrain,knntest,knntrain$ban\_id,k=1)  
#mean(knntest$ban\_id != KNN.pred)  
#table(KNN.pred, knntest$ban\_id)

#KNN.pred <- knn(knntrain,knntest,knntrain$ban\_id,k=3)  
#mean(knntest$ban\_id != KNN.pred)  
#table(KNN.pred, knntest$ban\_id)

KNN.pred <- knn(knntrain,knntest,knntrain$ban\_id,k=5)  
mean(knntest$ban\_id != KNN.pred)

## [1] 0.0252282

table(KNN.pred, knntest$ban\_id)

0 0 0 0 0 0 0 1106

#KNN.pred <- knn(knntrain,knntest,knntrain$ban\_id,k=7)  
#mean(knntest$ban\_id != KNN.pred)  
#table(KNN.pred, knntest$ban\_id)

#KNN.pred <- knn(knntrain,knntest,knntrain$ban\_id,k=15)  
#mean(knntest$ban\_id != KNN.pred)  
#table(KNN.pred, knntest$ban\_id)

Then we tried to make a usable function, in which we give the name of the champion we use, and we get the name of the champion we should ban.

#KNN.pred

predictions <- data.frame("Champion" = knntest$champion\_id,   
 "Ban\_prediction" = KNN.pred,   
 "True\_ban" = knntest$ban\_id )

lol\_champ[1,5]

## [1] "Wukong"

rownames(lol\_champ) <- lol\_champ$id  
lol\_champ <- lol\_champ[order(lol\_champ$id),]  
champ\_name <- function (id){  
 return(lol\_champ$name[lol\_champ$id == id])  
}  
result <- function(name){  
 return(lol\_champ$id[lol\_champ$name== name])  
}  
result('Darius')

## [1] 122

champ\_name(122) # our id translator

## [1] "Darius"

Suggestion <- function(name){  
 id <- result(name)  
 return(champ\_name(predictions$Ban\_prediction[predictions$Champion == id]))  
}  
Suggestion('Ashe') #Gives us an array of suggestions :D, don't pay attention to

## Warning in `==.default`(lol\_champ$id, id): longer object length is not a  
## multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of  
## shorter object length

## [1] "Tristana" "Camille" NA NA NA NA

# red text

1. Logistic Regression

Now that we selected a champion and banned a fearsome foe, let’s get to the in- game events. Which events will be the ones that will set us up for victory?

In order to find out, let’s make correlation test and look at the results. After finding out which data is more likely to impact the result of the game, we take into consideration those that have more than 30% correlation, p-values associated that are more than .05. The correlation values tell us how strongly each variable impacts the ending result, while a low p-value shows us that data is consistent enough to reject the null hypothesis.

cor(lol\_data[c(-1,-3)],lol\_data$t2\_win) > .3

## [,1]  
## winner TRUE  
## firstTower TRUE  
## firstInhibitor TRUE  
## firstBaron FALSE  
## firstDragon TRUE  
## firstRiftHerald FALSE  
## t1\_champ1id FALSE  
## t1\_champ1\_sum1 FALSE  
## t1\_champ1\_sum2 FALSE  
## t1\_champ2id FALSE  
## t1\_champ2\_sum1 FALSE  
## t1\_champ2\_sum2 FALSE  
## t1\_champ3id FALSE  
## t1\_champ3\_sum1 FALSE  
## t1\_champ3\_sum2 FALSE  
## t1\_champ4id FALSE  
## t1\_champ4\_sum1 FALSE  
## t1\_champ4\_sum2 FALSE  
## t1\_champ5id FALSE  
## t1\_champ5\_sum1 FALSE  
## t1\_champ5\_sum2 FALSE  
## t1\_towerKills FALSE  
## t1\_inhibitorKills FALSE  
## t1\_baronKills FALSE  
## t1\_dragonKills FALSE  
## t1\_riftHeraldKills FALSE  
## t1\_ban1 FALSE  
## t1\_ban2 FALSE  
## t1\_ban3 FALSE  
## t1\_ban4 FALSE  
## t1\_ban5 FALSE  
## t2\_champ1id FALSE  
## t2\_champ1\_sum1 FALSE  
## t2\_champ1\_sum2 FALSE  
## t2\_champ2id FALSE  
## t2\_champ2\_sum1 FALSE  
## t2\_champ2\_sum2 FALSE  
## t2\_champ3id FALSE  
## t2\_champ3\_sum1 FALSE  
## t2\_champ3\_sum2 FALSE  
## t2\_champ4id FALSE  
## t2\_champ4\_sum1 FALSE  
## t2\_champ4\_sum2 FALSE  
## t2\_champ5id FALSE  
## t2\_champ5\_sum1 FALSE  
## t2\_champ5\_sum2 FALSE  
## t2\_towerKills TRUE  
## t2\_inhibitorKills TRUE  
## t2\_baronKills TRUE  
## t2\_dragonKills TRUE  
## t2\_riftHeraldKills FALSE  
## t2\_ban1 FALSE  
## t2\_ban2 FALSE  
## t2\_ban3 FALSE  
## t2\_ban4 FALSE  
## t2\_ban5 FALSE  
## t2\_win TRUE

cor(lol\_data$firstRiftHerald,lol\_data$firstTower)

## [1] 0.1859889

vi <- sort(sample(1:nrow(lol\_data),.8\*nrow(lol\_data)))  
loltrain <- lol\_data[vi,]  
loltest <- lol\_data[-vi,]  
logic\_reg <- glm(t2\_win ~ firstInhibitor + firstTower + firstDragon+ t2\_towerKills + t2\_inhibitorKills + t1\_towerKills + t2\_baronKills + t1\_baronKills, data = loltrain, family = binomial)  
summary(logic\_reg)

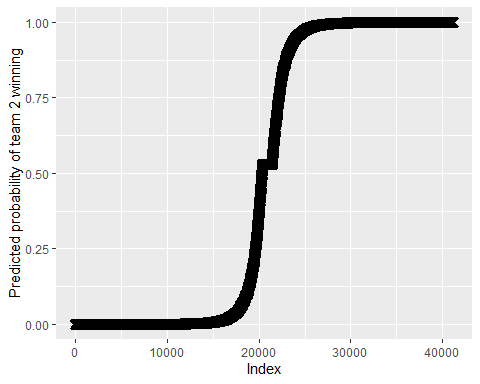
##   
## Call:  
## glm(formula = t2\_win ~ firstInhibitor + firstTower + firstDragon +   
## t2\_towerKills + t2\_inhibitorKills + t1\_towerKills + t2\_baronKills +   
## t1\_baronKills, family = binomial, data = loltrain)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.6841 -0.0345 -0.0019 0.0428 4.7589   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.12065 0.05965 2.023 0.04310 \*   
## firstInhibitor -0.63375 0.06533 -9.701 < 2e-16 \*\*\*  
## firstTower -0.71095 0.05906 -12.038 < 2e-16 \*\*\*  
## firstDragon 0.19059 0.05809 3.281 0.00104 \*\*   
## t2\_towerKills 1.35500 0.02779 48.762 < 2e-16 \*\*\*  
## t2\_inhibitorKills -0.30050 0.04233 -7.099 1.26e-12 \*\*\*  
## t1\_towerKills -1.06542 0.01896 -56.185 < 2e-16 \*\*\*  
## t2\_baronKills 0.35108 0.05991 5.861 4.61e-09 \*\*\*  
## t1\_baronKills -0.49032 0.06174 -7.942 1.99e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 57097.5 on 41191 degrees of freedom  
## Residual deviance: 7478.4 on 41183 degrees of freedom  
## AIC: 7496.4  
##   
## Number of Fisher Scoring iterations: 8

Here we plot a graph of the logistic regression function we made. If the predicted value was more than .5 (which gave us the better result when we look at the confusion matrix), then that means that team 2 is most likely a winner. Else, team 1 is.

logic\_prediction <- predict(logic\_reg, loltest, type = "response")  
modelingtest = ifelse(logic\_prediction < 0.5, "low", "high")   
table(modelingtest, loltest$t2\_win)

##   
## modelingtest 0 1  
## high 271 4935  
## low 4946 146

predicted\_data <- data.frame(prob\_win = logic\_reg$fitted.values, actual\_win = loltest$t2\_win)  
predicted\_data <- predicted\_data[order(predicted\_data$prob\_win,decreasing=FALSE),]  
predicted\_data$rank <- 1:nrow(predicted\_data)  
library(ggplot2)  
library(cowplot)  
ggplot(data= predicted\_data, aes(x = rank, y= prob\_win)) +  
 geom\_point(alpha=1,shape=4,stroke=2) +  
 xlab("Index") +  
 ylab("Predicted probability of team 2 winning")



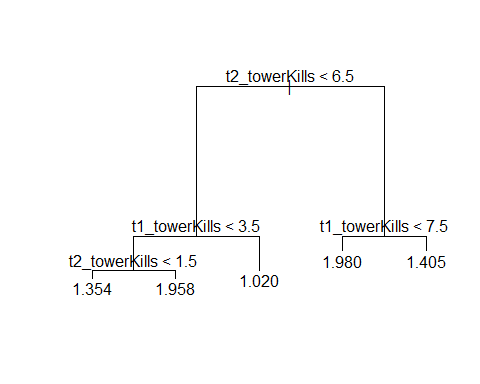
1. Decision Tree

library(tree)  
v = sample(1:nrow(lol\_data),0.8\*nrow(lol\_data))  
train <- sample(1:nrow(lol\_data),0.8\*nrow(lol\_data))  
test <- lol\_data[-train, "winner"]

tree.lol\_win <- tree(winner ~ .-t2\_win, lol\_data, subset = train)  
summary(tree.lol\_win)

##   
## Regression tree:  
## tree(formula = winner ~ . - t2\_win, data = lol\_data, subset = train)  
## Variables actually used in tree construction:  
## [1] "t2\_towerKills" "t1\_towerKills"  
## Number of terminal nodes: 5   
## Residual mean deviance: 0.03956 = 1629 / 41190   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.97960 -0.02006 -0.02006 0.00000 0.02041 0.97990

plot(tree.lol\_win)  
text(tree.lol\_win , pretty = 0)

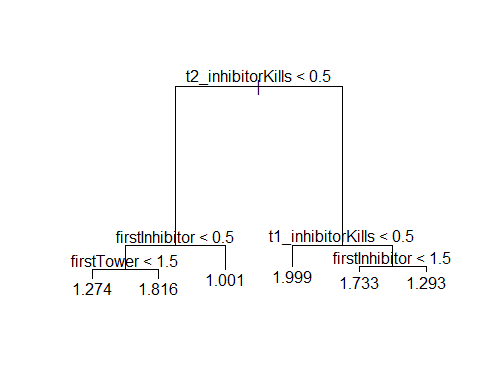
 Note: Numbers correspond to which team wins (ie. closer to 1, then team 1 wins closer to 2, then team 2 wins)

While it is no surprise that towerKills are the most important features, this is quite boring because you cannot win a game without destroying towers. So let’s remove towerKills and see what has an impact on the game outside of tower kills.

tree.lol\_win <- tree(winner ~ .-t2\_win-t2\_towerKills-t1\_towerKills,   
 lol\_data, subset = train)  
summary(tree.lol\_win)

##   
## Regression tree:  
## tree(formula = winner ~ . - t2\_win - t2\_towerKills - t1\_towerKills,   
## data = lol\_data, subset = train)  
## Variables actually used in tree construction:  
## [1] "t2\_inhibitorKills" "firstInhibitor" "firstTower"   
## [4] "t1\_inhibitorKills"  
## Number of terminal nodes: 6   
## Residual mean deviance: 0.04463 = 1838 / 41190   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.9990000 -0.0012330 -0.0012330 0.0000000 0.0009692 0.9988000

plot(tree.lol\_win)  
text(tree.lol\_win , pretty = 0)

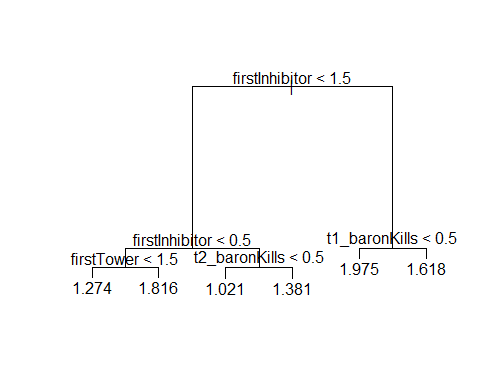
 Ok this is slightly more interesting, but a winning team MUST take at least 1 inhibitor AND 5 towers. The most interesting features to come out of this are firstTower & firstInhibitor (which correspond with which team took the first tower and first inhibitor), these are more interesting features because it indicates that going for the first tower or inhibitor could mean you are more likely to win. According to this tree, getting firstTower tower means you are more likely to win, but getting firstInhibitor means you are actually more likely to LOSE (this is somewhat coutnerintiuive, but likely due to inhibitors spawning minions which are harder to kill but provide more resources, which can allow comebacks to happen)

Let’s remove inhibitorKills because those are also needed in order to win (unless the enemy surrenders).

tree.lol\_win <- tree(winner ~ .-t2\_win-t2\_towerKills-t1\_towerKills  
 -t2\_inhibitorKills-t1\_inhibitorKills, lol\_data,   
 subset = train)  
summary(tree.lol\_win)

##   
## Regression tree:  
## tree(formula = winner ~ . - t2\_win - t2\_towerKills - t1\_towerKills -   
## t2\_inhibitorKills - t1\_inhibitorKills, data = lol\_data, subset = train)  
## Variables actually used in tree construction:  
## [1] "firstInhibitor" "firstTower" "t2\_baronKills" "t1\_baronKills"   
## Number of terminal nodes: 6   
## Residual mean deviance: 0.07644 = 3148 / 41190   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.97540 -0.02069 -0.02069 0.00000 0.02460 0.97930

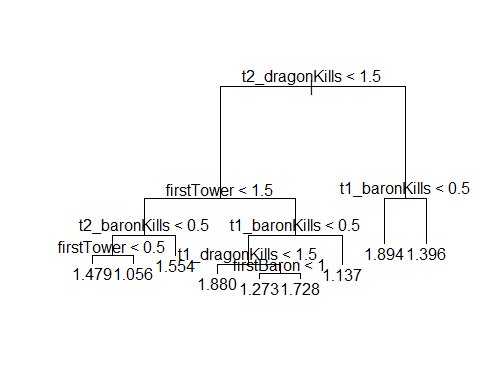
plot(tree.lol\_win)  
text(tree.lol\_win , pretty = 0)

 Awesome, so firstInhibitor seems to be important but so does baronKills and firstTower. Let’s push this one step forward and take out firstInhibitor to see the features buried below the usual game winning givens.

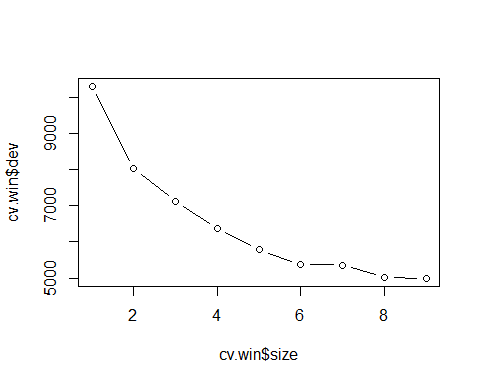
tree.lol\_win <- tree(winner ~ .-t2\_win-t2\_towerKills-t1\_towerKills  
 -t2\_inhibitorKills-t1\_inhibitorKills-firstInhibitor,   
 lol\_data, subset = train)  
summary(tree.lol\_win)

##   
## Regression tree:  
## tree(formula = winner ~ . - t2\_win - t2\_towerKills - t1\_towerKills -   
## t2\_inhibitorKills - t1\_inhibitorKills - firstInhibitor, data = lol\_data,   
## subset = train)  
## Variables actually used in tree construction:  
## [1] "t2\_dragonKills" "firstTower" "t2\_baronKills" "t1\_baronKills"   
## [5] "t1\_dragonKills" "firstBaron"   
## Number of terminal nodes: 9   
## Residual mean deviance: 0.1194 = 4919 / 41180   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.89360 -0.05568 -0.05568 0.00000 0.10640 0.94430

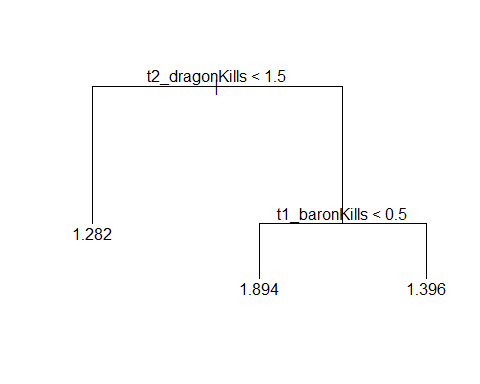
plot(tree.lol\_win)  
text(tree.lol\_win , pretty = 0)

 This is interesting! dragonKills shoots up beyond firstTower and baronKills in the decision tree, which makes it arguably more important than firstTower as an objective! Another interesting thing is that gameDuration finally comes into the decision tree, which means that closing out games early/holding on for late game is an important factor (but obviously less so than the game winning objectives)

cv.win <- cv.tree(tree.lol\_win)  
plot(cv.win$size, cv.win$dev, type = "b")

 6 is still the best (original), but 2/3 are surprisingly strong for how simple they are; so, let’s plot one below and see what it looks like.

prune.win <- prune.tree(tree.lol\_win , best = 3)  
plot(prune.win)  
text(prune.win , pretty = 0)

 3 returns a very accurate pruned tree, but the original longer tree of 6 terminal nodes is still the best tree (this is not a huge surprise because the original tree is small to begin with, but it is interesting to see how important firstInhibitor is a feature as it accounts for the vast majority of improvement of the model according to the pruning plot above)