Predicting VCT Matches

Saketh Marrapu

2024-12-28

Predicting Valorant Match Results with Logistic Regression

Reformatting the Data

Data Cleaning

Not all VCT events have all of the required data for each team. As such, these values were replaced with -1 when the data was being collected. In order to make this data usable we can make these values NA and remove them.

```
vct24dat <- read.csv("C:/Users/troll/Documents/valorantpredictive/rawvct2024data.csv",header = TRUE)
vct24dat[vct24dat==-1] = NA
vct24dat <- na.omit(vct24dat)
```

Reformatting factors

As the data is now, it's aggregate data for each team in each match. This, however is not useful since for instance the total number of kills changes based on the number of rounds which also changes based on whether a match is best of 3 or best of 5. As such, these variables should be averaged across the number of rounds. This is done for the Kills, Deaths, and Assists.

```
vct24dat$numRounds = vct24dat$FK + vct24dat$FD
vct24dat$KpR = vct24dat$KpR / vct24dat$numRounds
vct24dat$DpR = vct24dat$DpR / vct24dat$numRounds
vct24dat$ApR = vct24dat$ApR / vct24dat$numRounds
```

Some factors also need to be averaged for each player. For instance, the headshot percentage as an aggregate does not make sense since it is a sum of percentages. As such we should be averaging it across the number of players (5) in order to get the average headshot percentage of the team. We will do the same for KAST (Kill, Assist, Trade, Survive percent).

```
vct24dat$KAST = vct24dat$KAST / 5
vct24dat$HS = vct24dat$HS / 5
```

For the sake of consistency we shall do the same for Average Damage per Round (ADR) and Average Combat Score (ACS) so we don't have a mix of aggregate factors and factors averaged across players. This should not affect the regression because we are simply scaling these values by a scalar.

```r
vct24dat$ACS = vct24dat$ACS / 5
vct24dat$ADR = vct24dat$ADR / 5
```

We do still, however, find that we have aggregate values in First Kills (FK) and First Deaths (FD). These variables also depend on the number of rounds which of course changes based on the number of maps played in a match. However, we cannot average this based on the number of rounds as you can only have 1 first kill or first death each round, and averaging would suggest something like having "0.5 first kills per round and 0.6 first deaths per round". As such, I will turn this into a categorical variable, FKFD which is True if a team has a higher or equal amount of First Kills than First Deaths and False if a team has less First Kills than First Deaths.

```r
vct24dat$FKFD <- vct24dat$FK >= vct24dat$FD
```

Finally we'll convert the W/L stat into a boolean variable.

```r
vct24dat$W.L <- vct24dat$W.L == 1
```

Reformatting Columns

First, I will remove the useless columns. Since we converted the First Kill and First Death columns into a boolean, we don't need them anymore. In addition, the numRounds column was only needed for calculation purposes and isn't relevant to the regression.

```r
vct24dat <- vct24dat[,c(-8,-9,-11)]
vct24dat <- vct24dat[,c(1:7,9,8)]
finalDat <- vct24dat
```

This completes the reformatting of data. We are left with a data frame that has 640 rows (one for each match with data in the Valorant Champions Tour 2024) and 9 columns. The columns are as follows:
ACS: The average ACS of each player on a team during a match
KpR: The average kills per round of the team during a match
DpR: The average deaths per round of the team during a match
ApR: The average assists per round of the team during a match
KAST: The average percentage of rounds with kills, assists, survives, and trades per player on a team in a match
ADR: The average damage per round of each player on a team during a match
HS: The average headshot percentage of the team during a match
FKFD: Whether a team had greater or equal first kills than first deaths
W.L: Whether the team won or lost the match. This is the response variable.

Regression

First we must decide what factors to use in the model, we can do this through analyzing the AIC of the models. We prefer to use AIc here because R^2 adjusted doesn't work for logistic regression (it's based on linear model residuals) and BIC is attempting to model a true value from candidates. Since Valorant matches are not actually a model that is created from the given variables, BIC will not be useful in finding a model to predict future valorant matches. As such, AIC is more appropriate for model selection analysis in this case. In order to test the dataset in the future we will be using a data split, where we train the model on a subset of data while holding the rest to test the accuracy of the model.

```r
library(BMA)
library(caret)
set.seed(2133)
trainIndex <- createDataPartition(finalDat$W.L,p=0.8,list=FALSE)
trainData <- finalDat[trainIndex,]
testData <- finalDat[-trainIndex,]
output <- glm(W.L ~ ., data=finalDat,family="binomial")
summary(output)
```

```
##
## Call:
## glm(formula = W.L ~ ., family = "binomial", data = finalDat)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.77326    6.71869   0.264   0.7918
## ACS         -0.14266    0.05585  -2.554   0.0106 *
## KpR         10.71866    1.89686   5.651 1.60e-08 ***
## DpR         -7.69095    1.16595  -6.596 4.22e-11 ***
## ApR          0.86249    0.83120   1.038   0.2994
## KAST        -0.03268    0.06749  -0.484   0.6282
## ADR          0.12434    0.06597   1.885   0.0595 .
## HS           0.02853    0.05138   0.555   0.5787
## FKFDTRUE     0.07911    0.32557   0.243   0.8080
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 887.23  on 639  degrees of freedom
## Residual deviance: 271.25  on 631  degrees of freedom
## AIC: 289.25
##
## Number of Fisher Scoring iterations: 7
```

```r
step(output)
```

```
## Start:  AIC=289.25
## W.L ~ ACS + KpR + DpR + ApR + KAST + ADR + HS + FKFD
##
##        Df Deviance    AIC
## - FKFD  1   271.31 287.31
## - KAST  1   271.48 287.48
## - HS    1   271.56 287.56
## - ApR   1   272.34 288.34
## <none>      271.25 289.25
## - ADR   1   274.89 290.89
## - ACS   1   277.92 293.92
## - KpR   1   300.91 316.91
## - DpR   1   340.49 356.49
##
## Step:  AIC=287.31
## W.L ~ ACS + KpR + DpR + ApR + KAST + ADR + HS
##
##        Df Deviance    AIC
## - KAST  1   271.53 285.53
## - HS    1   271.64 285.64
## - ApR   1   272.39 286.39
## <none>      271.31 287.31
## - ADR   1   274.96 288.96
## - ACS   1   278.00 292.00
## - KpR   1   301.28 315.28
## - DpR   1   344.00 358.00
##
## Step:  AIC=285.53
## W.L ~ ACS + KpR + DpR + ApR + ADR + HS
##
##        Df Deviance    AIC
## - HS    1   271.82 283.82
## - ApR   1   272.44 284.44
## <none>      271.53 285.53
## - ADR   1   275.16 287.16
## - ACS   1   279.48 291.48
## - KpR   1   301.33 313.33
## - DpR   1   351.50 363.50
##
## Step:  AIC=283.82
## W.L ~ ACS + KpR + DpR + ApR + ADR
##
##        Df Deviance    AIC
## - ApR   1   272.50 282.50
## <none>      271.82 283.82
## - ADR   1   276.35 286.35
## - ACS   1   280.29 290.29
## - KpR   1   302.14 312.14
## - DpR   1   351.89 361.89
##
## Step:  AIC=282.5
## W.L ~ ACS + KpR + DpR + ADR
##
##        Df Deviance    AIC
## <none>      272.50 282.50
## - ADR   1   276.46 284.46
## - ACS   1   280.29 288.29
## - KpR   1   302.42 310.42
## - DpR   1   352.00 360.00
```

```
##
## Call:  glm(formula = W.L ~ ACS + KpR + DpR + ADR, family = "binomial",
##     data = finalDat)
##
## Coefficients:
## (Intercept)          ACS          KpR          DpR          ADR
##      0.7914      -0.1417      10.8555      -7.5285       0.1208
##
## Degrees of Freedom: 639 Total (i.e. Null);  635 Residual
## Null Deviance:       887.2
## Residual Deviance: 272.5     AIC: 282.5
```

Using AIC, we get a model with 4 variables: ACS, KpR, DpR, and ADR. This can be explained by the fact that many of the variables in the dataset are most likely highly correlated with each other. For instance, if a team has on average a higher headshot percentage that means that they will on average get more kills in less shots, thus increasing the amount of kills they would get in a round before they die themselves.

```
finalModel <- glm(formula = W.L ~ ACS + KpR + DpR + ADR, family = "binomial", data = finalDat)
summary(finalModel)
```

```
##
## Call:
## glm(formula = W.L ~ ACS + KpR + DpR + ADR, family = "binomial",
##     data = finalDat)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.79138    5.96849   0.133  0.89452
## ACS         -0.14173    0.05113  -2.772  0.00557 **
## KpR         10.85550    1.89129   5.740 9.48e-09 ***
## DpR         -7.52851    1.08083  -6.965 3.27e-12 ***
## ADR          0.12077    0.06123   1.972  0.04856 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 887.23  on 639  degrees of freedom
## Residual deviance: 272.50  on 635  degrees of freedom
## AIC: 282.5
##
## Number of Fisher Scoring iterations: 7
```

```
# we turn the predictions into a logical boolean vector based on whether the probability of winning is
predictions <- pnorm(scale(predict.glm(finalModel,testData[,-ncol(testData)],type="response"))) >= 0.5
proportionCorrect = 1 - abs((sum(predictions) - sum(testData$W.L)) / sum(testData$W.L))
proportionCorrect
```

```
## [1] 0.953125
```

As we can see, the model predicts the probability of a team winning a match with high accuracy. The proportion it got correct was 0.953125.

Using the Model

First let's import the off season data for sentinels and Nrg and do the same cleaning that we did for the vct 2024 data.

```r
treatData <- function(df){
  df[df==-1] = NA
  df <- na.omit(df)
  df$numRounds = df$FK + df$FD
  df$KpR = df$KpR / df$numRounds
  df$DpR = df$DpR / df$numRounds
  df$ApR = df$ApR / df$numRounds
  df$KAST = df$KAST / 5
  df$HS = df$HS / 5
  df$ACS = df$ACS / 5
  df$ADR = df$ADR / 5
  df$FKFD <- df$FK >= df$FD
  df$W.L <- df$W.L == 1
  df <- df[,c(-8,-9,-11)]
  df <- df[,c(1:7,9,8)]
  return(df)
}

senOffSeason <- read.csv("C:/Users/troll/Documents/valorantpredictive/sentinelsOffSeasonData.csv",heade
senOffSeason <- treatData(senOffSeason)
nrgOffSeason <- read.csv("C:/Users/troll/Documents/valorantpredictive/nrgOffSeasonData.csv",header=TRUE
nrgOffSeason <- treatData(nrgOffSeason)
```

After this, we can take the predicted probability of each team winning based on the data from each match. Averaging these predicted probabilities we can get the chance of them winning any given match for the future seasons.

```r
senProbability <- sum(pnorm(scale(predict.glm(finalModel,senOffSeason,type="response")))) / nrow(senOff
senProbability
```

```
## [1] 0.5035851
```

```r
nrgProbability <- sum(pnorm(scale(predict.glm(finalModel,nrgOffSeason,type="response")))) / nrow(nrgOff
nrgProbability
```

```
## [1] 0.5344107
```

As we can see, based on the off season, NRG has a higher probability of winning any given match (0.5344107) when compared to Sentinels (0.5035851) and as such in a match between NRG and Sentinels, NRG *should* win.