R Notebook

Code

Brandon Runyon 9/25/22

Data taken from Kaggle (https://www.kaggle.com/datasets/aliibrahim10/valorant-stats). Data taken from the leader board for the game VALORANT.

These models are built for single classification, but the ratings in the game are multiclass. This means I will have to do one-for-all to do the regression.

The rating system the game uses is similar to the Elo Rating System (https://en.wikipedia.org/wiki/Elo_rating_system). At the end of each match, it compares your performance to the other players in the game. Over time, the players approach the rating that best represents them.

I will use this data of overall statistics for the top players during a single season of the game to try to predict their rating.

Load and preprocess

Load “val_stats.csv”

Hide

```
df <- read.csv("val_stats.csv")
str(df)
```

```
'data.frame':	85678 obs. of  38 variables:
 $ region          : chr  NA NA NA NA ...
 $ name            : chr  "ShimmyXD" "XSET Cryo" "PuRelittleone" "Boba" ...
 $ tag             : chr  "#NA1" "#cells" "#yoruW" "#0068" ...
 $ rating          : chr  "Radiant" "Radiant" "Radiant" "Radiant" ...
 $ damage_round    : num  136 170 148 178 150 ...
 $ headshots       : chr  "992" "879" "720" "856" ...
 $ headshot_percent: num  24.9 28.3 24 37.3 24.4 26 25.2 17.5 24.6 20.8 ...
 $ aces            : int  0 2 3 3 2 2 7 2 2 1 ...
 $ clutches        : int  140 122 117 83 71 162 186 112 189 56 ...
 $ flawless        : int  80 94 59 49 38 94 92 64 132 44 ...
 $ first_bloods    : chr  "161" "316" "216" "235" ...
 $ kills           : chr  "1,506" "1,608" "1,115" "1,134" ...
 $ deaths          : chr  "1,408" "1,187" "1,064" "812" ...
 $ assists         : chr  "703" "206" "267" "157" ...
 $ kd_ratio        : num  1.07 1.35 1.05 1.4 1.11 1.03 1.16 1.17 1.31 1.37 ...
 $ kills_round     : num  0.7 1 0.8 1 0.8 0.7 0.9 0.8 0.9 0.9 ...
 $ most_kills      : int  29 32 39 37 29 33 37 36 38 29 ...
 $ score_round     : num  209 271 228 277 231 ...
 $ wins            : int  59 52 42 32 32 57 69 44 88 29 ...
 $ win_percent     : num  59.6 65.8 65.6 62.8 62.8 58.2 55.6 62 56.8 69 ...
 $ agent_1         : chr  "Fade" "Chamber" "Yoru" "Jett" ...
 $ agent_2         : chr  "Viper" "Jett" "Jett" "Chamber" ...
 $ agent_3         : chr  "Omen" "Raze" "Chamber" "KAY/O" ...
 $ gun1_name       : chr  "Vandal" "Vandal" "Vandal" "Vandal" ...
 $ gun1_head       : int  35 41 38 51 36 40 35 26 8 29 ...
 $ gun1_body       : int  59 56 57 47 60 55 61 67 91 66 ...
 $ gun1_legs       : int  5 3 4 2 4 5 4 7 1 5 ...
 $ gun1_kills      : chr  "802" "689" "444" "754" ...
 $ gun2_name       : chr  "Phantom" "Operator" "Phantom" "Sheriff" ...
 $ gun2_head       : int  33 8 36 48 21 35 32 6 36 2 ...
 $ gun2_body       : int  62 91 61 51 71 62 64 93 59 98 ...
 $ gun2_legs       : int  5 0 3 1 8 3 5 1 5 1 ...
 $ gun2_kills      : chr  "220" "226" "231" "48" ...
 $ gun3_name       : chr  "Classic" "Phantom" "Operator" "Phantom" ...
 $ gun3_head       : int  36 32 8 44 8 29 48 35 22 23 ...
 $ gun3_body       : int  60 63 91 56 92 65 50 65 69 70 ...
 $ gun3_legs       : int  3 5 1 0 0 6 2 0 9 6 ...
 $ gun3_kills      : int  147 137 102 36 64 135 253 100 34 85 ...
```

Check the number of NAs in each column.

Hide

```
sapply(df, function(x) sum(is.na(x)))
```

```
         region            name             tag          rating     damage_round        head
shots headshot_percent            aces         clutches
          20865               0               0               0                0
0                0               0               0
       flawless     first_bloods           kills          deaths          assists         kd_
ratio      kills_round       most_kills      score_round
              0               0               0               0                0
0                0               0               0
           wins      win_percent         agent_1         agent_2          agent_3        gun1
_name        gun1_head        gun1_body        gun1_legs
              0               0               0               0                0
0                0               0               0
      gun1_kills        gun2_name        gun2_head        gun2_body        gun2_legs       gun2_
kills        gun3_name        gun3_head        gun3_body
              0               0               0               0                0
0                0               0               0
       gun3_legs       gun3_kills
              0               0
```

NA in region refers to North America, so no NA data, but needs to be handled for factors.

Convert columns to factors

Hide

```
cols <- c("region", "rating", "agent_1", "agent_2", "agent_3", "gun1_name", "gun2_name", "gun3_n
ame")
df[cols] <- lapply(df[cols], factor, exclude = NULL)
```

Some columns are read as strings instead of numbers due to commas.

Hide

```
cols <- c("headshots", "first_bloods", "kills", "deaths", "assists", "gun1_kills", "gun2_kills")
df[cols] <- lapply(df[cols], gsub, pattern = ",", replacement = "")
df[cols] <- lapply(df[cols], as.numeric)
```

View new data

Hide

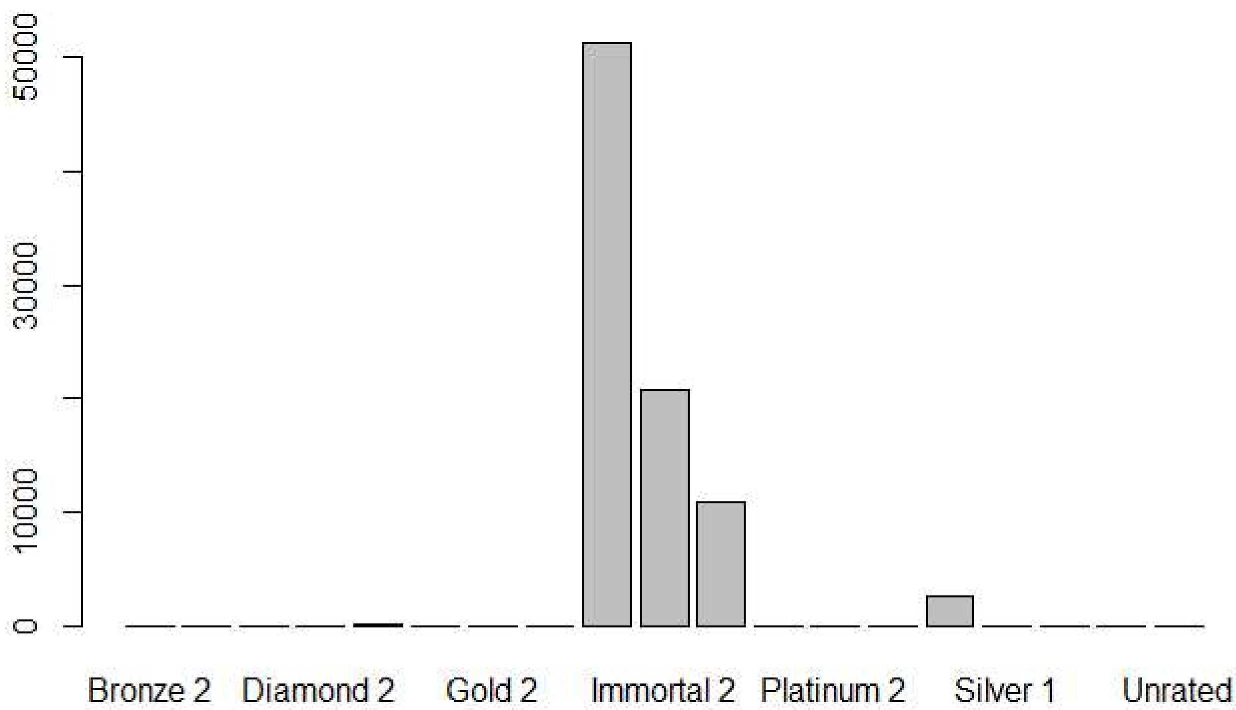
```
str(df)
```

```
'data.frame':	85678 obs. of  38 variables:
 $ region         : Factor w/ 6 levels "AP","BR","EU",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ name           : chr  "ShimmyXD" "XSET Cryo" "PuRelittleone" "Boba" ...
 $ tag            : chr  "#NA1" "#cells" "#yoruW" "#0068" ...
 $ rating         : Factor w/ 19 levels "Bronze 2","Bronze 3",..: 15 15 15 15 15 15 15 15 15 15 
...
 $ damage_round   : num  136 170 148 178 150 ...
 $ headshots      : num  992 879 720 856 534 ...
 $ headshot_percent: num  24.9 28.3 24 37.3 24.4 26 25.2 17.5 24.6 20.8 ...
 $ aces           : int  0 2 3 3 2 2 7 2 2 1 ...
 $ clutches       : int  140 122 117 83 71 162 186 112 189 56 ...
 $ flawless       : int  80 94 59 49 38 94 92 64 132 44 ...
 $ first_bloods   : num  161 316 216 235 137 179 311 215 515 103 ...
 $ kills          : num  1506 1608 1115 1134 869 ...
 $ deaths         : num  1408 1187 1064 812 781 ...
 $ assists        : num  703 206 267 157 213 629 614 341 440 140 ...
 $ kd_ratio       : num  1.07 1.35 1.05 1.4 1.11 1.03 1.16 1.17 1.31 1.37 ...
 $ kills_round    : num  0.7 1 0.8 1 0.8 0.7 0.9 0.8 0.9 0.9 ...
 $ most_kills     : int  29 32 39 37 29 33 37 36 38 29 ...
 $ score_round    : num  209 271 228 277 231 ...
 $ wins           : int  59 52 42 32 32 57 69 44 88 29 ...
 $ win_percent    : num  59.6 65.8 65.6 62.8 62.8 58.2 55.6 62 56.8 69 ...
 $ agent_1        : Factor w/ 19 levels "Astra","Breach",..: 6 4 19 7 7 6 7 7 4 4 ...
 $ agent_2        : Factor w/ 20 levels "","Astra","Breach",..: 19 8 8 5 14 18 7 5 8 14 ...
 $ agent_3        : Factor w/ 20 levels "","Astra","Breach",..: 12 14 5 9 5 5 9 18 9 16 ...
 $ gun1_name      : Factor w/ 16 levels "Ares","Bucky",..: 16 16 16 16 16 16 16 16 11 12 ...
 $ gun1_head      : int  35 41 38 51 36 40 35 26 8 29 ...
 $ gun1_body      : int  59 56 57 47 60 55 61 67 91 66 ...
 $ gun1_legs      : int  5 3 4 2 4 5 4 7 1 5 ...
 $ gun1_kills     : num  802 689 444 754 419 ...
 $ gun2_name      : Factor w/ 18 levels "Ares","Bucky",..: 13 12 13 14 16 13 13 12 18 12 ...
 $ gun2_head      : int  33 8 36 48 21 35 32 6 36 2 ...
 $ gun2_body      : int  62 91 61 51 71 62 64 93 59 98 ...
 $ gun2_legs      : int  5 0 3 1 8 3 5 1 5 1 ...
 $ gun2_kills     : num  220 226 231 48 65 144 369 318 626 163 ...
 $ gun3_name      : Factor w/ 18 levels "Ares","Bucky",..: 4 13 12 13 12 16 14 9 13 16 ...
 $ gun3_head      : int  36 32 8 44 8 29 48 35 22 23 ...
 $ gun3_body      : int  60 63 91 56 92 65 50 65 69 70 ...
 $ gun3_legs      : int  3 5 1 0 0 6 2 0 9 6 ...
 $ gun3_kills     : int  147 137 102 36 64 135 253 100 34 85 ...
```

Most data is from the top few players (those labeled with Immortal or Radiant).

Hide

```
counts <- table(df$rating)
barplot(counts)
```



Remove all players who aren't Immortal 1/2/3, or Radiant and refactor.

Hide

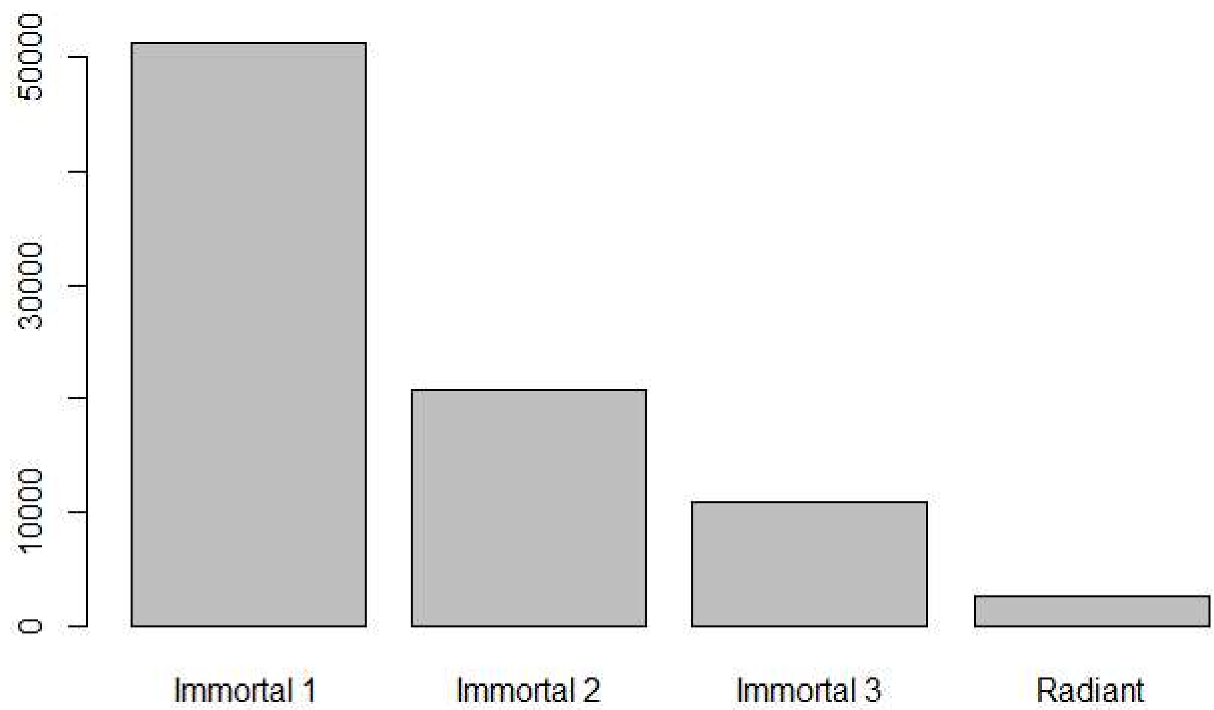
```
i <- which(df$rating == "Immortal 1" | df$rating == "Immortal 2" | df$rating == "Immortal 3" | d
f$rating == "Radiant")
df <- df[i,]
df$rating <- factor(df$rating)
str(df)
```

```
'data.frame':   85574 obs. of  38 variables:
 $ region          : Factor w/ 6 levels "AP","BR","EU",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ name            : chr  "ShimmyXD" "XSET Cryo" "PuRelittleone" "Boba" ...
 $ tag             : chr  "#NA1" "#cells" "#yoruW" "#0068" ...
 $ rating          : Factor w/ 4 levels "Immortal 1","Immortal 2",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ damage_round    : num  136 170 148 178 150 ...
 $ headshots       : num  992 879 720 856 534 ...
 $ headshot_percent: num  24.9 28.3 24 37.3 24.4 26 25.2 17.5 24.6 20.8 ...
 $ aces            : int  0 2 3 3 2 2 7 2 2 1 ...
 $ clutches        : int  140 122 117 83 71 162 186 112 189 56 ...
 $ flawless        : int  80 94 59 49 38 94 92 64 132 44 ...
 $ first_bloods    : num  161 316 216 235 137 179 311 215 515 103 ...
 $ kills           : num  1506 1608 1115 1134 869 ...
 $ deaths          : num  1408 1187 1064 812 781 ...
 $ assists         : num  703 206 267 157 213 629 614 341 440 140 ...
 $ kd_ratio        : num  1.07 1.35 1.05 1.4 1.11 1.03 1.16 1.17 1.31 1.37 ...
 $ kills_round     : num  0.7 1 0.8 1 0.8 0.7 0.9 0.8 0.9 0.9 ...
 $ most_kills      : int  29 32 39 37 29 33 37 36 38 29 ...
 $ score_round     : num  209 271 228 277 231 ...
 $ wins            : int  59 52 42 32 32 57 69 44 88 29 ...
 $ win_percent     : num  59.6 65.8 65.6 62.8 62.8 58.2 55.6 62 56.8 69 ...
 $ agent_1         : Factor w/ 19 levels "Astra","Breach",..: 6 4 19 7 7 6 7 7 4 4 ...
 $ agent_2         : Factor w/ 20 levels "","Astra","Breach",..: 19 8 8 5 14 18 7 5 8 14 ...
 $ agent_3         : Factor w/ 20 levels "","Astra","Breach",..: 12 14 5 9 5 5 9 18 9 16 ...
 $ gun1_name       : Factor w/ 16 levels "Ares","Bucky",..: 16 16 16 16 16 16 16 16 11 12 ...
 $ gun1_head       : int  35 41 38 51 36 40 35 26 8 29 ...
 $ gun1_body       : int  59 56 57 47 60 55 61 67 91 66 ...
 $ gun1_legs       : int  5 3 4 2 4 5 4 7 1 5 ...
 $ gun1_kills      : num  802 689 444 754 419 ...
 $ gun2_name       : Factor w/ 18 levels "Ares","Bucky",..: 13 12 13 14 16 13 13 12 18 12 ...
 $ gun2_head       : int  33 8 36 48 21 35 32 6 36 2 ...
 $ gun2_body       : int  62 91 61 51 71 62 64 93 59 98 ...
 $ gun2_legs       : int  5 0 3 1 8 3 5 1 5 1 ...
 $ gun2_kills      : num  220 226 231 48 65 144 369 318 626 163 ...
 $ gun3_name       : Factor w/ 18 levels "Ares","Bucky",..: 4 13 12 13 12 16 14 9 13 16 ...
 $ gun3_head       : int  36 32 8 44 8 29 48 35 22 23 ...
 $ gun3_body       : int  60 63 91 56 92 65 50 65 69 70 ...
 $ gun3_legs       : int  3 5 1 0 0 6 2 0 9 6 ...
 $ gun3_kills      : int  147 137 102 36 64 135 253 100 34 85 ...
```

Same table now shows the top players only.

Hide

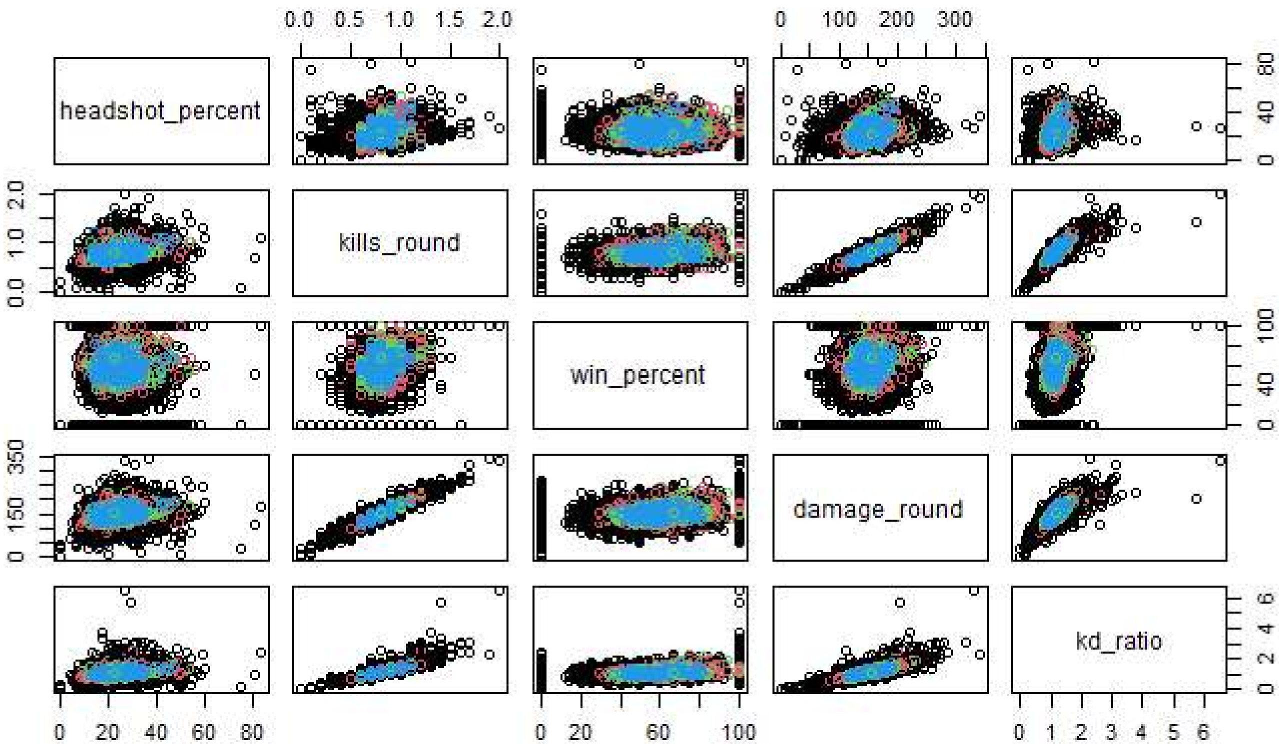
```
counts <- table(df$rating)
barplot(counts)
```



Basic analysis of the data

Hide

```
pairs(df[,c("headshot_percent","kills_round","win_percent", "damage_round", "kd_ratio")], col=df
$rating)
```



Win percent has several with 0% or 100% win rate, which is unrealistic. These are likely players who only played the 1 required game to get their rank for the season.

These data values are also very tightly packed regardless of rating. They only seem to get tighter by rating.

Hide

```
i <- which(df$win_percent == 0 | df$win_percent == 100)
df <- df[-i,]
```

Multiple possible ratings, so we must divide it for multiclass

Hide

```
dfR <- df
dfR$rating <- as.factor(ifelse(dfR$rating=="Radiant",1,0))

dfI1 <- df
dfI1$rating <- as.factor(ifelse(dfI1$rating=="Immortal 1",1,0))

dfI2 <- df
dfI2$rating <- as.factor(ifelse(dfI2$rating=="Immortal 2",1,0))

dfI3 <- df
dfI3$rating <- as.factor(ifelse(dfI3$rating=="Immortal 3",1,0))
```

Logistic Regression

Define function

Hide

```
fun <- function(df, i){
  train <- df[i,]
  test <- df[-i,]
  glm1 <- glm(rating~win_percent+headshot_percent+kd_ratio+kills_round+damage_round+win_percent*
headshot_percent*kd_ratio*kills_round*damage_round, data=train, family="binomial")
  probs <- predict(glm1, newdata=test)
  pred <- ifelse(probs>0.5, 1, 0)
  acc <- mean(pred==test$rating)
  print(paste("accuracy = ", acc))
  table(pred, test$rating)
}
```

Run for Radiant

Hide

```
set.seed(1)
i <- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
fun(dfR, i)
```

```
[1] "accuracy =  0.96841101442365"

pred     0     1
   0 16248   530
```

Run for Immortal 1

Hide

```
fun(dfI1, i)
```

```
[1] "accuracy =  0.589223983788294"

pred    0    1
   0 5105 5182
   1 1710 4781
```

Run for Immortal 2

Hide

```
fun(dfI2, i)
```

```
[1] "accuracy =  0.755989986887591"

pred     0     1
   0 12684  4094
```

Run for Immortal 3

Hide

```
fun(dfI3, i)
```

```
[1] "accuracy =  0.869352723804983"

pred     0     1
   0 14586  2191
   1     1     0
```

These tests did not work very well with this data set. The largest difference between the ranks is that the data is tighter around a central value. A higher rating means the player is more likely to remain near the average. This can be seen in the pairs graph above as most points are in the same area. The model ended up just guessing false for everything since that gave it the highest accuracy.

Naive Bayes

Create function for naive bayes

Hide

```
library(e1071)
fun1 <- function(df, i){
  train <- df[i,]
  test <- df[-i,]
  nb <- naiveBayes(rating~win_percent+headshot_percent+kd_ratio+kills_round+damage_round, data=t
rain)
  print(nb)
  pred <- predict(nb, newdata=test, type="class")
  acc <- mean(pred==test$rating)
  print(paste("accuracy = ", acc))
  table(pred, test$rating)
}
```

Run for Radiant

Hide

```
fun1(dfR, i)
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
         0          1 
0.96905035 0.03094965 

Conditional probabilities:
   win_percent
Y       [,1]     [,2]
  0 53.56631 9.138226
  1 56.06399 7.220316

   headshot_percent
Y       [,1]     [,2]
  0 23.53090 5.073874
  1 25.34651 4.980761

   kd_ratio
Y       [,1]      [,2]
  0 1.050496 0.1421023
  1 1.101555 0.1191619

   kills_round
Y        [,1]       [,2]
  0 0.7569120 0.09273009
  1 0.7886856 0.07938866

   damage_round
Y       [,1]     [,2]
  0 142.6366 15.60404
  1 147.3625 13.26784

[1] "accuracy =  0.96841101442365"

pred     0     1
   0 16248   530
   1     0     0
```

Run for Immortal 1

Hide

```
fun1(dfI1, i)
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.4072628 0.5927372 

Conditional probabilities:
   win_percent
Y       [,1]     [,2]
  0 55.79711 8.190080
  1 52.16397 9.386057

   headshot_percent
Y       [,1]     [,2]
  0 24.19726 5.022495
  1 23.16786 5.078089

   kd_ratio
Y       [,1]      [,2]
  0 1.072925 0.1269254
  1 1.037752 0.1493776

   kills_round
Y        [,1]       [,2]
  0 0.7686912 0.08378162
  1 0.7504777 0.09736658

   damage_round
Y       [,1]     [,2]
  0 144.3245 14.04007
  1 141.7237 16.43764

[1] "accuracy =  0.615448802002623"

pred    0    1
   0 3546 3183
   1 3269 6780
```

Run for Immortal 2

Hide

```
fun1(dfI2, i)
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.7539227 0.2460773 

Conditional probabilities:
   win_percent
Y       [,1]     [,2]
  0 52.89024 9.137343
  1 55.95178 8.561876

   headshot_percent
Y       [,1]     [,2]
  0 23.55021 5.117249
  1 23.70010 4.965579

   kd_ratio
Y       [,1]      [,2]
  0 1.047421 0.1451864
  1 1.066338 0.1295099

   kills_round
Y        [,1]       [,2]
  0 0.7561360 0.09469698
  1 0.7632857 0.08523610

   damage_round
Y       [,1]     [,2]
  0 142.5264 15.92215
  1 143.5687 14.35906

[1] "accuracy =  0.755989986887591"

pred     0     1
   0 12684  4094
   1     0     0
```

Run for Immortal 3

Hide

```
fun1(dfI3, i)
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.8697641 0.1302359 

Conditional probabilities:
   win_percent
Y       [,1]    [,2]
  0 53.37441 9.26101
  1 55.44145 7.66064

   headshot_percent
Y       [,1]     [,2]
  0 23.39597 5.062470
  1 24.86350 5.017937

   kd_ratio
Y       [,1]      [,2]
  0 1.048110 0.1439635
  1 1.078568 0.1225014

   kills_round
Y        [,1]       [,2]
  0 0.7554609 0.09388165
  1 0.7741533 0.08091141

   damage_round
Y       [,1]     [,2]
  0 142.4463 15.82086
  1 145.0306 13.46434

[1] "accuracy =  0.869352723804983"

pred     0     1
   0 14586  2191
   1     1     0
```

Similarly to the logistic regression, this model didn’t work very well. It ended up just saying false for everything except Immortal 1.

Conclusion

Both of these models didn’t work for this data set. They were unable to accurately distinguish between the data points since they were so tightly packed.

Because a players rank changes based on individual games rather than overall statistics, it makes it more difficult to accurately predict using these models.