Final Project Stat 310 APPENDIX

Heewon Oh 301268860

09/12/2021

This is a full Appendix including all code that was used for the Project.

```
library(tidyverse)
data <- read_csv("games.csv")
```

Part A: Setting up the first dataset

```
library(dplyr)
library(tidyverse)
data1 <- data%>%
  mutate(rated = as.integer(rated))%>%
  #mutate(draw = if_else(victory_status==1,1,0))%>%
  #mutate(victory_status = as.integer(victory_status))%>%
  mutate(winner=as.factor(winner))%>%
  mutate(winner = as.integer(winner))%>%
  #mutate(whitewinner = if_else(winner==3,1,0))%>%
  filter(winner !=2)%>%
  mutate(winner = if_else(winner==3, 1,0))%>%
  mutate(victory_status = as.factor(victory_status))%>%
  mutate(mate = if_else(victory_status=="mate",1,0))%>%
  mutate(resign = if_else(victory_status=="resign",1,0))%>%
  mutate(outoftime=if_else(victory_status=="outoftime",1,0))%>%
  dplyr::select(rated,turns,white_rating,black_rating,opening_ply,mate,resign,outoftime,winner)


set.seed(310)
#data2 <- data1[sample(1:nrow(data1), size = 5000, replace=F),]

index = sample(1:nrow(data1), round(0.7*nrow(data1)), replace=F)
train = data1[index,]
test = data1[-index,]
dim(train)
```

```
## [1] 13376     9
```

```
dim(test)
```

```
## [1] 5732    9
```

```r
train_x = train[, -9]
train_y = train[,9]

test_x = test[, -9]
test_y = test[,9]

apply(data1, 2, function(x) sum( is.na(x) ) )
```

```
##        rated        turns white_rating black_rating  opening_ply         mate
##            0            0            0            0            0            0
##       resign    outoftime       winner
##            0            0            0
```

Part B: Descriptive Stats Generation

```r
library(qwraps2)
options(qwraps2_markup = "markdown")
Chess_Data<-data1
statstable <- as.data.frame(Chess_Data)
summary_statistics <-
  list(
    "Turns" =
      list(
        "mean (sd)" = ~qwraps2::mean_sd(turns, na_rm = TRUE),
        "min" = ~min(turns, na.rm = TRUE),
        "max" = ~max(turns, na.rm = TRUE)
      ),
    "Rated" =
      list(
        "mean (sd)" = ~qwraps2::mean_sd(rated, na_rm = TRUE),
        "min" = ~min(rated, na.rm = TRUE),
        "max" = ~max(rated, na.rm = TRUE)
      ),
    "White Rating" =
      list(
        "mean (sd)" = ~qwraps2::mean_sd(white_rating, na_rm = TRUE),
        "min" = ~min(white_rating, na.rm = TRUE),
        "max" = ~max(white_rating, na.rm = TRUE)

      ),
      "Black Rating" =
      list(
        "mean (sd)" = ~qwraps2::mean_sd(black_rating, na_rm = TRUE),
        "min" = ~min(black_rating, na.rm = TRUE),
        "max" = ~max(black_rating, na.rm = TRUE)

      ),
      "Opening Play" =
      list(
        "mean (sd)" = ~qwraps2::mean_sd(opening_ply, na_rm = TRUE),
        "min" = ~min(opening_ply, na.rm = TRUE),
```

```
        "max" = ~max(opening_ply, na.rm = TRUE)
      ),
      "Winner" =
      list(
        "mean (sd)" = ~qwraps2::mean_sd(winner, na_rm = TRUE),
        "min" = ~min(winner, na.rm = TRUE),
        "max" = ~max(winner, na.rm = TRUE)
      )

    )
summary_table(Chess_Data, summary_statistics)
```

	Chess_Data (N = 19,108)
Turns	
mean (sd)	59.19 ± 32.31
min	1
max	349
Rated	
mean (sd)	0.81 ± 0.39
min	0
max	1
White Rating	
mean (sd)	1,593.70 ± 289.95
min	784
max	2700
Black Rating	
mean (sd)	1,586.23 ± 290.15
min	789
max	2723
Opening Play	
mean (sd)	4.81 ± 2.78
min	1
max	28
Winner	
mean (sd)	0.52 ± 0.50
min	0
max	1

Part C: Failed Neural Net Method Attempt

```
library(neuralnet)
nnet = neuralnet(winner~., train, hidden = c(8,3), linear.output = FALSE)
plot(nnet,rep="best")
```

```
ypred = neuralnet::compute(nnet, test[,-5])
yhat = ypred$net.result
head(yhat)
```

```r
c("black","white")[yhat[1,]==max(yhat[1,])]
sp = c("white","black")
tt = apply(yhat,1,function(x) sp[x==max(x)])
tt
```

```r
yclass = apply(yhat,1,function(x) sp[x==max(x)])
#yclass= sp[ apply(yhat, 1, which.max) ]
table(test_y$winner, yclass)
```

Logistic Regression

```r
mod = glm(winner~., data = train, family = binomial)
summary(mod)
```

```
##
## Call:
## glm(formula = winner ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.832  -1.103   0.441   1.063   3.081
##
## Coefficients: (1 not defined because of singularities)
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1363742  0.1376203   0.991 0.321712
## rated        -0.0170497  0.0498214  -0.342 0.732188
## turns        -0.0020619  0.0006043  -3.412 0.000645 ***
## white_rating  0.0038563  0.0001079  35.727  < 2e-16 ***
## black_rating -0.0039190  0.0001082 -36.224  < 2e-16 ***
## opening_ply   0.0137334  0.0070348   1.952 0.050911 .
## mate          0.1728158  0.0699486   2.471 0.013488 *
## resign        0.1067188  0.0672693   1.586 0.112639
## outoftime            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 18516  on 13375  degrees of freedom
## Residual deviance: 16399  on 13368  degrees of freedom
## AIC: 16415
##
## Number of Fisher Scoring iterations: 4
```

```r
mod.probs = predict(mod, test, type = "response")
#round(rbind(mod.probs[1:300]),3)

mod.pred = if_else(mod.probs>0.5,1,0)

table(mod.pred,test_y$winner)
```

```
##
## mod.pred    0    1
##        0 1587  833
##        1 1132 2180
```

```r
mean(mod.pred==test_y$winner)
```

```
## [1] 0.6571877
```

```r
mod2 = glm(winner~white_rating+black_rating+turns, data = train, family = binomial)
summary(mod2)
```

```
##
## Call:
## glm(formula = winner ~ white_rating + black_rating + turns, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8199  -1.1065   0.4475   1.0627   3.0287
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.2769426  0.1166707   2.374  0.01761 *
## white_rating  0.0038678  0.0001069  36.175  < 2e-16 ***
## black_rating -0.0039122  0.0001075 -36.403  < 2e-16 ***
## turns        -0.0020426  0.0005850  -3.492  0.00048 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 18516  on 13375  degrees of freedom
## Residual deviance: 16409  on 13372  degrees of freedom
## AIC: 16417
##
## Number of Fisher Scoring iterations: 4
```

```r
mod2.probs = predict(mod, test, type = "response")
#round(rbind(mod.probs[1:300]),3)

mod2.pred = if_else(mod.probs>0.5,1,0)

table(mod2.pred,test_y$winner)
```

```
##
## mod2.pred    0    1
##         0 1587  833
##         1 1132 2180
```

```r
log.acc1 <- mean(mod2.pred==test_y$winner)
log.acc1
```

```
## [1] 0.6571877
```

```r
log.prec1 <- (2180)/(2180+833)
log.rec1 <- 2180/(2180+1132)
```

```r
mod3 = glm(winner~white_rating+black_rating+turns+mate, data = train, family = binomial)
summary(mod2)
```

```
##
## Call:
## glm(formula = winner ~ white_rating + black_rating + turns, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8199  -1.1065   0.4475   1.0627   3.0287
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.2769426  0.1166707   2.374  0.01761 *
## white_rating  0.0038678  0.0001069  36.175  < 2e-16 ***
## black_rating -0.0039122  0.0001075 -36.403  < 2e-16 ***
## turns        -0.0020426  0.0005850  -3.492  0.00048 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 18516  on 13375  degrees of freedom
## Residual deviance: 16409  on 13372  degrees of freedom
## AIC: 16417
##
## Number of Fisher Scoring iterations: 4
```

```r
mod3.probs = predict(mod, test, type = "response")
#round(rbind(mod.probs[1:300]),3)

mod3.pred = if_else(mod.probs>0.5,1,0)

table(mod3.pred,test_y$winner)
```

```
##
## mod3.pred    0    1
##         0 1587  833
##         1 1132 2180
```

```r
log.acc2 <- mean(mod3.pred==test_y$winner)
log.acc2
```

```
## [1] 0.6571877
```

```r
log.prec2 <- (2180)/(2180+833)
log.rec2 <- 2180/(2180+1132)
```

LDA

```r
library(MASS)
lda.fit = lda(winner~black_rating+white_rating+turns, data = train)
lda.pred = predict(lda.fit, test)
lda.class = lda.pred$class
table(lda.class,test$winner)
```

```
##
## lda.class    0    1
##         0 1562  808
##         1 1157 2205
```

```r
lda.acc1 <- mean(lda.class == test$winner)
lda.prec1 <- (2205)/(2205+808)
lda.rec1 <- 2205/(2205+1157)
```

```r
lda.fit = lda(winner~black_rating+white_rating+turns+mate, data = train)
lda.pred = predict(lda.fit, test)
lda.class = lda.pred$class
table(lda.class,test$winner)
```

```
##
## lda.class    0    1
##         0 1555  805
##         1 1164 2208
```

```r
lda.acc2 <- mean(lda.class == test$winner)
lda.acc2
```

```
## [1] 0.6564899
```

```r
lda.prec2 <- (2208)/(2208+805)
lda.rec2 <- 2208/(2208+1164)
```

QDA

```r
qda.fit = qda(winner~white_rating+black_rating+turns, data = train)
qda.pred = predict(qda.fit, test)
qda.class = qda.pred$class
table(qda.class,test$winner)
```

```
##
## qda.class    0    1
##         0 1625  882
##         1 1094 2131
```

```r
qda.acc1 <- mean(qda.class == test$winner)
qda.acc1
```

```
## [1] 0.6552687
```

```r
qda.prec1 <- (2131)/(2131+882)
qda.rec1 <- 2131/(2131+1094)
```

```r
qda.fit = qda(winner~white_rating+black_rating+turns+mate, data = train)
qda.pred = predict(qda.fit, test)
qda.class = qda.pred$class
table(qda.class,test$winner)
```

```
##
## qda.class    0    1
##         0 1643  917
##         1 1076 2096
```

```r
qda.acc2<- mean(qda.class == test$winner)
qda.acc2
```

```
## [1] 0.6523029
```

```r
qda.prec2 <- (2096)/(2096+917)
qda.rec2 <- 2096/(2096+1076)
```

NB

```r
library(e1071)
nb.fit = naiveBayes(winner~white_rating+black_rating+turns, data = train)
nb.class = predict(nb.fit, test)
table(nb.class,test$winner)
```

```
##
## nb.class    0    1
##        0 1232  585
##        1 1487 2428
```

```r
nb.preds = predict(nb.fit, test, type = "raw")
nb.acc1<- mean(test$winner == nb.class)
nb.acc1
```

```
## [1] 0.6385206
```

```r
nb.prec1 <- (2428)/(2428+585)
nb.rec1 <- 2428/(2428+1487)
```

```r
nb.fit = naiveBayes(winner~white_rating+black_rating+turns+mate, data = train)
nb.class = predict(nb.fit, test)
table(nb.class,test$winner)
```

```
##
## nb.class    0    1
##        0 1234  601
##        1 1485 2412
```

```r
nb.preds = predict(nb.fit, test, type = "raw")
nb.acc2 <- mean(test$winner == nb.class)
nb.acc2
```

```
## [1] 0.6360782
```

```r
nb.prec2 <- (2412)/(2412+601)
nb.rec2 <- 2412/(2412+1485)
```

Part D: Creating Second dataset for tree-based methods

```r
library(plyr)
datat2 <- read_csv("games.csv")%>%
  filter(winner != "draw")%>%
  mutate(white_rating = round_any(white_rating, 50))%>%
  mutate(black_rating = round_any(black_rating, 50))%>%
  mutate(turns = round_any(turns, 5))%>%
  dplyr::select(rated, turns, victory_status, winner, white_rating, black_rating, opening_ply)
datat3 <-as.data.frame(datat2)
cnames = colnames(datat3)
datat3[cnames] = lapply(datat3[cnames], as.factor)


set.seed(310)
index = sample(1:nrow(datat3),nrow(datat3)*0.7,replace=F)
train2 = datat3[index,]
test2 = datat3[-index,]

table(train2$winner)/nrow(train2)
```

```
##
##     black     white
## 0.4776075 0.5223925
```

```r
table(test2$winner)/nrow(test2)
```

```
##
##     black     white
## 0.4742718 0.5257282
```

Part E: Failed Tree Method due to number of levels.

```r
tree = tree(winner ~ ., data = train2)

# prediction on train/test


print("======= On training =======")
tree.pred.trt = predict(tree,train, type="class")
tree.acc.trn = mean(tree.pred.trt==train2$winner)
tree.acc.trn
table(train2$winner,tree.pred.trt)

print("======= On testing =======")
tree.pred.tst = predict(tree,test, type="class")
tree.acc.tst = mean(tree.pred.tst==test2$winner)
tree.acc.tst
table(test2$winner,tree.pred.tst)
```

Bagging

```r
library(randomForest)
set.seed(310)

bag = randomForest(winner ~ .,
                   data = train2,
                   mtry = 6,
                   type="classification",
                   importance = TRUE,
                   ntrees = 500)
bag
```

```
##
## Call:
##  randomForest(formula = winner ~ ., data = train2, mtry = 6, type = "classification",      importanc
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 35.93%
## Confusion matrix:
##       black white class.error
## black  3281  3107   0.4863807
## white  1698  5289   0.2430228
```

```r
bag.acc.trn = mean(train2$winner==predict(bag,newdata=train2))
table(train2$winner,predict(bag,newdata=train2))
```

```
##
##         black white
##   black  6358    30
##   white   313  6674
```

```r
bag.acc.trn
```

```
## [1] 0.9743551
```

```r
bag.pred.tst = predict(bag,newdata=test2)
table(test2$winner,bag.pred.tst)
```

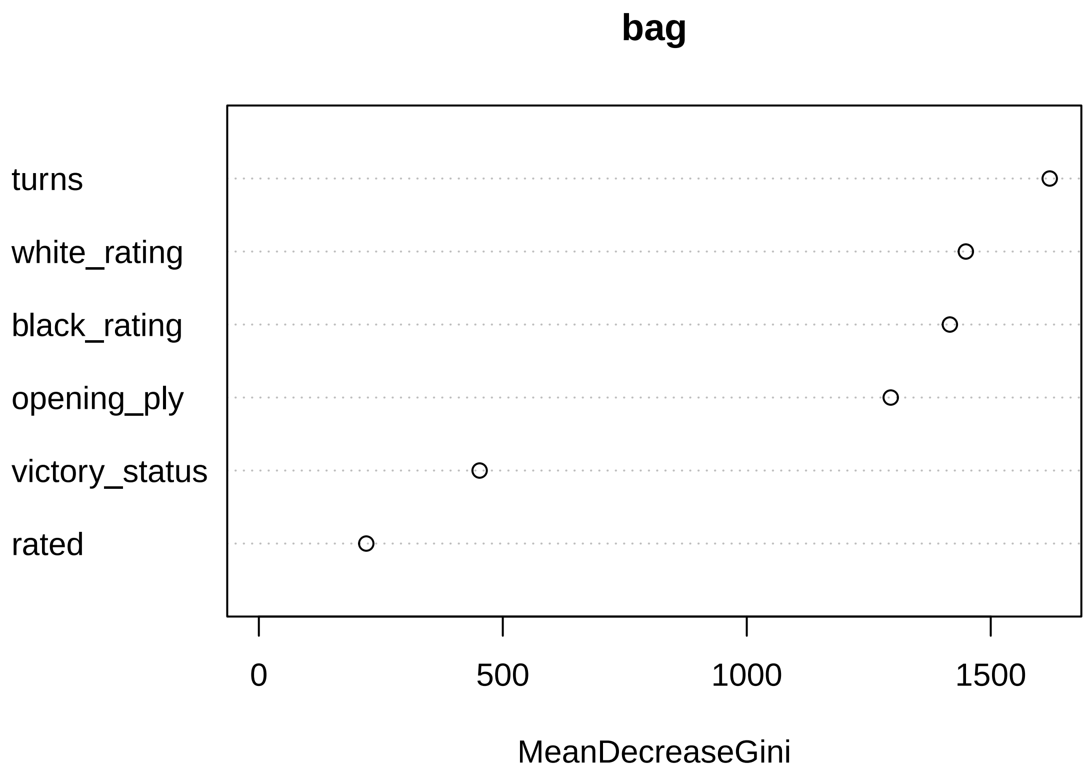
```
##        bag.pred.tst
##         black white
##   black  1401  1318
##   white   735  2279
```

```r
bag.acc.tst = mean(test2$winner==bag.pred.tst)
bag.acc.tst
```

```
## [1] 0.6418978
```

```r
bag.prec1 <- (2279)/(2279+1318)
bag.rec1 <- 2279/(2279+735)
```

```r
varImpPlot(bag, type=2)
```



Random Forest

```
rdf = randomForest(winner ~ .,
                   data = train2,
                   mtry = 2,
                   type="classification",
                   importance = TRUE,
                   ntrees = 500)
rdf
```

```
##
## Call:
##  randomForest(formula = winner ~ ., data = train2, mtry = 2, type = "classification",      importanc
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 36.04%
## Confusion matrix:
##       black white class.error
## black  2947  3441   0.5386662
## white  1380  5607   0.1975097
```

```r
rdf.acc.trn = mean(train2$winner==predict(rdf,newdata=train2))
table(train2$winner,predict(rdf,newdata=train2))
```

```
##
##          black white
##    black  6255   133
##    white   248  6739
```

```r
rdf.acc.trn
```

```
## [1] 0.971514
```

```r
rdf.pred.tst = predict(rdf,newdata=test2)
table(test2$winner,rdf.pred.tst)
```

```
##         rdf.pred.tst
##          black white
##    black  1273  1446
##    white   607  2407
```

```r
rdf.acc.tst = mean(test2$winner==rdf.pred.tst)
rdf.acc.tst
```

```
## [1] 0.6418978
```

```r
rdf.prec1 <- (2407)/(2407+1446)
rdf.rec1 <- 2407/(2407+607)
```

Failed KNN

```r
library(caret)
library(class)
class::predict = knn(train = train_x,
             test = test_x,
             cl = train_y,
             k = 19)
error.rate = mean(test_y !=predict)
error.rate
```

Boost Method

```r
library(gbm)
seat.trn <- train2
seat.trn$winner= ifelse(seat.trn$winner=="black",0,1)

boost = gbm(winner ~ .,
```

```r
            data = seat.trn,
            distribution = "bernoulli",
            n.trees = 5000,
            interaction.depth = 4,
            shrinkage = 0.01)
boost
```

```
## gbm(formula = winner ~ ., distribution = "bernoulli", data = seat.trn,
##     n.trees = 5000, interaction.depth = 4, shrinkage = 0.01)
## A gradient boosted model with bernoulli loss function.
## 5000 iterations were performed.
## There were 6 predictors of which 5 had non-zero influence.
```

```r
boost.pred.tst = ifelse(predict(boost, test2, n.trees = 5000, "response") > 0.5, "white", "black")
table(test2$winner,boost.pred.tst)
```

```
##        boost.pred.tst
##         black white
##   black  1671  1048
##   white   917  2097
```

```r
boost.acc.tst = mean(test2$winner==boost.pred.tst)
boost.acc.tst
```

```
## [1] 0.6572475
```

```r
boost.prec1 <- (2097)/(2097+1048)
boost.rec1 <- 2097/(2097+917)
```

Part F: Results Table

```r
cbind( method=c("Logistic", "Log w Mate", "LDA","LDA w Mate","QDA","QDA w Mate","Naive Baines","NB w Ma
      accuracy = round(c(log.acc1, log.acc2,lda.acc2, qda.acc1,qda.acc2,nb.acc1,nb.acc2,bag.acc.tst,rdf
      precision = round(c(log.prec1,log.prec2,lda.prec1,lda.prec2,qda.prec1,qda.prec2,nb.prec1,nb.prec2
      recall = round(c(log.rec1,log.rec2,lda.rec1,lda.rec2,qda.rec1,qda.rec2,nb.rec1,nb.rec2,bag.rec1,r
)
```

```
##       method          accuracy  precision recall
##  [1,] "Logistic"      "0.65719" "0.72353" "0.65821"
##  [2,] "Log w Mate"    "0.65719" "0.72353" "0.65821"
##  [3,] "LDA"           "0.65649" "0.73183" "0.65586"
##  [4,] "LDA w Mate"    "0.65527" "0.73282" "0.6548"
##  [5,] "QDA"           "0.6523"  "0.70727" "0.66078"
##  [6,] "QDA w Mate"    "0.63852" "0.69565" "0.66078"
##  [7,] "Naive Baines"  "0.63608" "0.80584" "0.62018"
##  [8,] "NB w Mate"     "0.6419"  "0.80053" "0.61894"
##  [9,] "Bagging"       "0.6419"  "0.63358" "0.75614"
## [10,] "Random Forest" "0.65725" "0.62471" "0.79861"
## [11,] "Boosting"      "0.65719" "0.66677" "0.69575"
```