R Code supporting "Modeling All-NBA Team Voting" project

```
##### Data Loading and Prep #####

bball <- read.csv(file=paste(path,"allnbawithstatus.csv", sep=""))
bball <- bball[,-c(1, 2, 4, 6, 32, 33, 34, 35, 36)]
bball <- bball[,-c(41,46,53,54,55,56)]
bball <- bball[,-28]
bball$ALLNBA=as.factor(bball$ALLNBA)
bball$Pos <- as.factor(bball$Pos)
bball$thiseason <- as.factor(bball$thiseason)

colnames(bball)[colnames(bball) == "Player_x"] ="Player"
colnames(bball)[colnames(bball) == "Age_x"] ="Age"
colnames(bball)[colnames(bball) == "G_x"] ="Gms"
colnames(bball)[colnames(bball) == "GS"] ="GmsStarted"
colnames(bball)[colnames(bball) == "MP_x"] ="MP"
colnames(bball)[colnames(bball) == "FG."] ="FGPct"
colnames(bball)[colnames(bball) == "X3P."] ="X3PPct"
colnames(bball)[colnames(bball) == "X2P."] ="X2PPct"
colnames(bball)[colnames(bball) == "eFG."] ="EffFGPct"
colnames(bball)[colnames(bball) == "FT."] ="FTPct"
colnames(bball)[colnames(bball) == "TS."] ="TSPct"
colnames(bball)[colnames(bball) == "ORB."] ="ORBPct"
colnames(bball)[colnames(bball) == "DRB."] ="DRBPct"
colnames(bball)[colnames(bball) == "TRB."] ="TRBPct"
colnames(bball)[colnames(bball) == "AST."] ="ASTPct"
colnames(bball)[colnames(bball) == "STL."] ="STLPct"
colnames(bball)[colnames(bball) == "BLK."] ="BLKPct"
colnames(bball)[colnames(bball) == "TOV."] ="TOVPct"
colnames(bball)[colnames(bball) == "USG."] ="USGPct"
colnames(bball)[colnames(bball) == "WS.48"] ="WSper48"
colnames(bball)[colnames(bball) == "thiseason"] ="Season"

bball[is.na(bball)]<-0

mod_log <- glm(ALLNBA ~ ., data = bball[,c(50,3:47)], family=binomial(link='logit'))
cd<-cooks.distance(mod_log)
cdo<-cd[cd>=1]
cdo
plot(cd)

bball<-bball[bball$Gms>10,]

basic<-bball[bball$Season != "2022",c(50, 48, 1, 49, 2:47)]
recentyear<-bball[bball$Season == "2022",c(50, 48, 1, 49, 2:47)]

library(glmnet)
library(dplyr)
library(class)
library(rpart)
library(randomForest)

##### 10 FOLD CROSS VALIDATION #####

set.seed(2929)

TEALL = NULL
lassolambdatracker = NULL
knnktracker = NULL
pcaknnpcktracker = NULL


basicmix<-basic[sample(nrow(basic)),]
folds = cut(seq(1,nrow(basicmix)), breaks=10, labels=FALSE)


for(i in 1:10){
  testflag <- which(folds==i, arr.ind = TRUE)
  bbtest=basicmix[testflag, ]
  bbtrain=basicmix[-testflag, ]


  ##### Logistic Regression #####

  mod_log <- glm(ALLNBA ~ ., data = bbtrain[,c(1,5:50)], family=binomial(link='logit'))
```

```
  ## Testing Error
  pred_log <- predict(mod_log, bbtest[,-1], type="response")
  pred_log_rounded <- ifelse(pred_log >= 0.50, 1, 0)
  TestErr_log <- NULL
  TestErr_log <- c(TestErr_log, mean(pred_log_rounded != bbtest$ALLNBA))


  ##### LASSO #####

  x <- as.matrix(bbtrain[,5:50])
  mod_lasso <- glmnet(x, y=bbtrain[,1], alpha=1, family="binomial")

  cv.glmmod <- cv.glmnet(x, y=as.numeric(bbtrain[,1]), alpha=1)

  best.lambda <- cv.glmmod$lambda.1se

  lasso.coef_all <- coef(mod_lasso, s=best.lambda)
  lasso.coef <- lasso.coef_all[-1,1]
  lasso.int  <- lasso.coef_all[1,1]

  pred_lasso <- as.matrix(bbtest[,5:50]) %*% as.vector(lasso.coef) + lasso.int
  pred_lasso_rounded <- ifelse(pred_lasso > 0, 1, 0)

  TestErr_lasso <- NULL
  TestErr_lasso <- c(TestErr_lasso, mean(pred_lasso_rounded != bbtest$ALLNBA));

  lassolambdatracker <- rbind(lassolambdatracker, cbind(best.lambda, TestErr_lasso))


  ##### KNN #####

  std_train <- bbtrain[,5:50] %>% mutate_all(~(scale(.) %>% as.vector))   # Scaling because KNN is distance-
based
  std_test <- bbtest[,5:50] %>% mutate_all(~(scale(.) %>% as.vector))

  ## Testing Error
  TestErr_knn <- NULL
  knn_k <- NULL
  for (kk in c(3,5,7,9,11,13,15)){
    pred_knn <- knn(std_train, std_test, bbtrain[,1], k=kk)
    knn_k <- rbind(knn_k, kk)
    TestErr_knn <- rbind(TestErr_knn, mean(pred_knn != bbtest[,1]))
  }

  k.opt <- which.min(TestErr_knn)

  pred_knn <- knn(std_train, std_test, bbtrain[,1], k=k.opt*2+1) #Preds with KNN and best k
  TestErr_knn <- NULL
  TestErr_knn <- c(TestErr_knn, mean(pred_knn != bbtest$ALLNBA));

  knnktracker <- rbind(knnktracker, cbind(k.opt*2+1, TestErr_knn))


  ##### PCA w KNN #####

  mod_pca <- prcomp(bbtrain[,5:50], scale=TRUE)

  pcacum <- summary(mod_pca)$importance[3,]
  pc.opt = which(pcacum==pcacum[pcacum >= 0.8][1])

  pca_df<-data.frame(mod_pca$x[,1:pc.opt])
  pca_df$ALLNBA<-bbtrain$ALLNBA

  pca_df_test <- as.data.frame(predict(mod_pca, bbtest[,5:50]))
  pca_df_test$ALLNBA <- bbtest$ALLNBA

  TestErr_pcaknn <- NULL
  pcaknn_k <- NULL
  for (kk in c(3,5,7,9,11,13,15)){
    pred_pcaknn <- knn(pca_df[,1:pc.opt], pca_df_test[,1:pc.opt], pca_df$ALLNBA, k=kk)
    pcaknn_k <- rbind(pcaknn_k, kk)
    TestErr_pcaknn <- rbind(TestErr_pcaknn, mean(pred_pcaknn != bbtest[,1]))
  }

  k.opt <- which.min(TestErr_knn)

  pred_pcaknn<-knn(pca_df[,1:pc.opt], pca_df_test[,1:pc.opt], pca_df$ALLNBA, k=k.opt*2+1)
  TestErr_pcaknn <- NULL
```

```
  TestErr_pcaknn <- c(TestErr_pcaknn, mean(pred_pcaknn != bbtest$ALLNBA));

  pcaknnpcktracker <- rbind(pcaknnpcktracker, cbind(pc.opt, k.opt*2+1, TestErr_pcaknn))


  ##### Classification Tree #####

  mod_tree <- rpart(ALLNBA~., method = "class", data = bbtrain[,c(1,5:50)])

  pred_tree<-predict(mod_tree, bbtest, type="class")
  TestErr_tree <- NULL
  TestErr_tree <- c(TestErr_tree, mean(pred_tree != bbtest$ALLNBA));

  TEALL <- rbind(TEALL, cbind(TestErr_log, TestErr_lasso, TestErr_knn, TestErr_pcaknn, TestErr_tree))

}

dim(TEALL);

colnames(TEALL) <- c("LOG", "LASSO", "KNN", "PCAKNN", "TREE")

MeanTE<-apply(TEALL, 2, mean);
print(MeanTE)

#LOG        LASSO       KNN         PCAKNN      TREE
#0.01507984 0.01702631 0.01629653 0.02031024 0.02152531

lassolambdatracker
knnktracker
pcaknnpcktracker


##### Random Forest #####

RFTE<-NULL
print(timestamp())
for(i in 1:10){
  testflag <- which(folds==i, arr.ind = TRUE)
  bbtest=basicmix[testflag, ]
  bbtrain=basicmix[-testflag, ]

  for(mtryn in c(5,7,9,11,13)){
    for(ntreen in c(501,751,1001,1251)){

      mod_RF_tun = randomForest(x = bbtrain[5:50], y = as.factor(bbtrain$ALLNBA), ntree = ntreen, mtry=mtryn)

      pred_RF_tun <- predict(mod_RF_tun, bbtest[5:50])
      TestErr_RF_tun <- NULL
      TestErr_RF_tun <- c(TestErr_RF_tun, mean(pred_RF_tun != bbtest$ALLNBA));

      RFTE <- rbind(RFTE, cbind(mtryn, ntreen, TestErr_RF_tun))
    }
  }
}
print(timestamp())

RF_TEDF<-as.data.frame(RFTE)
RF_TEDF['Parameters']=paste('mtry=',RF_TEDF$mtryn,'; ntree=',RF_TEDF$ntreen)

RF_Results <- RF_TEDF %>% group_by(Parameters) %>% summarise(meanTE=mean(TestErr_RF_tun))

library(ggplot2)

ggplot(RF_Results, aes(x=Parameters, y=meanTE, label=round(meanTE,4))) +
  geom_bar(stat="identity") + coord_flip() + geom_text(size=3, hjust=1.5, color='white') +
  ggtitle('Mean 10-Fold CV Training Error by Parameter Options - Random Forest')
# mtry = 13; ntree = 1001; training error = 0.0152


##### Boosting - FIT #####

library(gbm)

trainboost <- basicmix[,c(1,5:50)]
trainboost$ALLNBA <- as.numeric(basicmix$ALLNBA)-1

mod_gbm <- gbm(ALLNBA ~ .,data=trainboost,
               distribution = 'bernoulli',
               n.trees = 5000,
```

```
               shrinkage = 0.01,
               interaction.depth = 2,
               cv.folds = 10)

## Find the estimated optimal number of iterations
perf_gbm = gbm.perf(mod_gbm, method="cv")  # Cross validation without my explicit loops
perf_gbm  # 2251

## Which variances are important
summary(mod_gbm)  #VORP PER WS

pred_gbm <- predict(mod_gbm, newdata = trainboost[-1], n.trees=perf_gbm, type="response")
pred_gbm_rounded <- ifelse(pred_gbm < 0.5, 0, 1)
TestErr_gbm <- NULL
TestErr_gbm <- c(TestErr_gbm, mean(pred_gbm_rounded != trainboost$ALLNBA))
TestErr_gbm  # 0.0050


##### Boosting - ASSESS 2021-2022 ALL NBA SELECTIONS #####

currentboost <- recentyear[,c(1,5:50)]
currentboost$ALLNBA <- as.numeric(recentyear$ALLNBA)-1

Final_pred_gbm <- predict(mod_gbm, newdata = currentboost[-1], n.trees=perf_gbm, type="response")
Final_pred_gbm_rounded <- ifelse(Final_pred_gbm < 0.5, 0, 1)
FinalTE_gbm <- mean(Final_pred_gbm_rounded != currentboost$ALLNBA)
FinalTE_gbm  # 0.008097166

recentyear[recentyear$ALLNBA==1,c(4,3)]  # Actual Voting

recentyear[Final_pred_gbm_rounded==1,c(4,3)] # Model predictions

Final_pred_gbm[Final_pred_gbm_rounded==1]  # Actual percentages

Final_pred_gbm[recentyear['Player']=='Chris Paul' | recentyear['Player'] == 'Pascal Siakam']  # Predictions for
the ALL-NBA players the model didn't pick

excluded=recentyear[recentyear['Player']=='Chris Paul' | recentyear['Player'] == 'Pascal Siakam',]
excluded
excludedpred <- predict(mod_gbm, newdata=excluded[-1], n.trees = perf_gbm, type="response")
excludedpred
```