dec520Q-C-team49_Team_Project_Rcode

```
####################
#Data Cleaning
#####################
shot=read.csv('shot_logs.csv')
        #shot clock
shot$SHOT_CLOCK[is.na(shot$SHOT_CLOCK)] <- 0
shot$SHOT_CLOCK_NA=ifelse(is.na(shot$SHOT_CLOCK),1,0)
shot$SHOT_CLOCK_24=ifelse(shot$SHOT_CLOCK==24.00,"24 sec","not 24 sec")
shot$SHOT_CLOCK_24[is.na(shot$SHOT_CLOCK_24)] <- "0 sec"
        #location
shot$LOCATION=ifelse(shot$LOCATION=="A",0,1)
        #touch time
shot<-subset(shot,shot$TOUCH_TIME>=0)
        #game clock
library(lubridate)
shot$GAME_CLOCK=period_to_seconds(ms(shot$GAME_CLOCK))

#Checking wins and points related columns to remove remove redundancy
result_check1=unique(paste(shot$SHOT_RESULT,shot$FGM))
#SHOT_RESULT=FGM
#--- we can drop SHOT_RESULT
#result_check1
# "made 1"   "missed 0"

result_check2=unique(paste(shot$PTS_TYPE,shot$PTS,shot$FGM))
#result_check2
#"2 2 1" "3 0 0" "2 0 0" "3 3 1"
#PTS=PTS_TYPE when FGM=1 ELSE 0 (i.e. PTS_TYPE*FGM)
#---Consistent:Keep PTS for later use

#Outliers in defender distance

#Distribution of closest defender distance
hist(shot$CLOSE_DEF_DIST,
    main = "Distribution of Closest Defender Distance",
    xlab="Closest Defender Distance (ft)")
#majority within 10 feet --- how many records b/w 10-20ft and >20ft
sub_10=subset(shot,shot$CLOSE_DE4F_DIST>10)
#4236 records
sub_10to20=subset(subset(shot,shot$CLOSE_DEF_DIST>10),CLOSE_DEF_DIST<=20)
#4152 records
sub_20=subset(shot,shot$CLOSE_DEF_DIST>20)
#174 records
```

```
#Distribution of closest defender distance >10feet away (possible outliers)
hist(sub_10$CLOSE_DEF_DIST,
    main = "Distribution of Closest Defender Distance",
    xlab="Closest Defender Distance (ft)")
hist(sub_10to20$CLOSE_DEF_DIST,
    main = "Distribution of Closest Defender Distance",
    xlab="Closest Defender Distance (ft)")
hist(sub_20$CLOSE_DEF_DIST,
    main = "Distribution of Closest Defender Distance",
    xlab="Closest Defender Distance (ft)")
#most seem to be covered within 25 feet
sub_25=subset(shot,shot$CLOSE_DEF_DIST>25)
#64 records --- maybe use 25 as threshold if not 20

#View(sub_25)

#Outliers based on sht distance & Creating additional Shot related columns
#1.Open vs. Guarded (1,0)
shot$open_shots=ifelse(shot$CLOSE_DEF_DIST>6, 1, 0)

#2.Pass vs. Dribble Shot (1,0)
shot$pass_shots=ifelse(shot$DRIBBLES==0, 1, 0)

#3.Checking point outliers based on shot distance

#What is the min distance for a 3pt: NBA=22--need to cross verify
shots_check=subset(shot,PTS==3)
shots_check1=subset(shots_check,SHOT_DIST<21)
summary(shots_check1)
hist(shots_check1$SHOT_DIST,
    main = "Distribution of Outlier 3 Pointers(<21ft) based on Shot Distance",
    xlab="Shot Distance (ft)")
#Even with a leeway of 1 foot there are 45 rows with misrecorded SHOT_DIST
#for 3 pointers --- small percentage of dataset,drop rows

#What is the min distance for a 2pt: NBA=25ft max--need to cross verify
shots_check2=subset(shot,PTS==2)
shots_check3=subset(shots_check,SHOT_DIST>27)
summary(shots_check3)
hist(shots_check3$SHOT_DIST,
    main = "Distribution of Outlier 2 Pointers(>27ft) based on Shot Distance",
    xlab="Shot Distance (ft)")

#Corner3 Shot Column (1)
shot$corner_shots=ifelse(shot$PTS==3,
                          ifelse(shot$SHOT_DIST<=23, 1, 0), 0)
shots_chk=subset(shot,corner_shots==1)

#####################
#JOINING TABLES
#####################
library(dplyr)
library(ggplot2)
library(RColorBrewer)

players <- read.csv('players.csv')

players$Pos <- ifelse(players$Pos=='PG', 1, ifelse(players$Pos=='SG', 2,
ifelse(players$Pos=='SF', 3, ifelse(players$Pos=='PF', 4, 5))))

#only data for DEFENDER
defender <- select(players, Player_2, Pos, Age, Weight, Experience, Salary, Height_cm)
#change column names
colnames(defender)[colnames(defender)=="Pos"] <- "pos_def"
colnames(defender)[colnames(defender)=="Age"] <- "age_def"
colnames(defender)[colnames(defender)=="Weight"] <- "weight_def"
colnames(defender)[colnames(defender)=="Experience"] <- "experience_def"
colnames(defender)[colnames(defender)=="Salary"] <- "salary_def"
colnames(defender)[colnames(defender)=="Height_cm"] <- "height_def"


join1 <- left_join(shot, defender, by = c("CLOSEST_DEFENDER"="Player_2"))

#only data for SHOT TAKER
shot_taker <- select(players, Player, Pos, Age, Weight, Experience, Salary, Height_cm,
Country)
#change column names
colnames(shot_taker)[colnames(shot_taker)=="Pos"] <- "pos"
colnames(shot_taker)[colnames(shot_taker)=="Age"] <- "age"
colnames(shot_taker)[colnames(shot_taker)=="Weight"] <- "weight"
colnames(shot_taker)[colnames(shot_taker)=="Experience"] <- "experience"
colnames(shot_taker)[colnames(shot_taker)=="Salary"] <- "salary"
colnames(shot_taker)[colnames(shot_taker)=="Height_cm"] <- "height"
colnames(shot_taker)[colnames(shot_taker)=="Country"] <- "country"

join2 <- left_join(join1, shot_taker, by = c("player_name"="Player"))

join2$intl_plyr <- ifelse(join2$country == 'us',1,0)

#database going forward is join2
```

```
#######################################
#Data cleaning again and creating columns
#######################################
join2$age<-as.numeric(levels(join2$age))[join2$age]
join2$height<-as.numeric(levels(join2$height))[join2$height]
join2$age_def<-as.numeric(levels(join2$age_def))[join2$age_def]
join2$height_def<-as.numeric(levels(join2$height_def))[join2$height_def]

join2$W <- ifelse(join2$W == 'W',1,0)
join2 <- na.omit(join2)

###################################
#Data Exploration-Correlation analysis
####################################
library(corrplot)

m <- cor(join2[,c(3-11)])#,13-15,20-34)])
corrplot(m)

correlations_mat <- select(join2, FINAL_MARGIN, SHOT_NUMBER, GAME_CLOCK,
SHOT_CLOCK, DRIBBLES, TOUCH_TIME, SHOT_DIST, CLOSE_DEF_DIST, open_shots,
height_def, salary_def, height, salary)
m <- cor(correlations_mat)
corrplot(m)

####################################
#Data Exploration-Distribution of Shots
####################################
shot <- read.csv('shot_logs.csv')

head(shot)

players <- read.csv('players.csv')

#round down values
shot$SHOT_CLOCK2 <- floor(shot$SHOT_CLOCK)
shot$SHOT_DIST2 <- floor(shot$SHOT_DIST)

#------------------------------

attach(shot)
aggdata <-aggregate(shot, by=list(SHOT_CLOCK2,SHOT_DIST2),
                    FUN=mean, na.rm=TRUE)
num <-shot %>% filter(SHOT_CLOCK2>=0) %>% count(SHOT_DIST2, SHOT_CLOCK2)
base2 <-cbind(aggdata,num[,3])
base2<-subset(base2, base2$n>1)
```

```
t2 <- ggplot(base2, aes(SHOT_CLOCK2,SHOT_DIST2))
#t2 + geom_point(aes(size=n),shape=15, fill=PTS)
t2 + geom_point(aes(color=PTS, size=n), shape=15)+
  scale_colour_gradientn(colours = terrain.colors(10)) +
  xlab('Shot Clock') +
  ylab('Shot Distance') +
  ggtitle('Points per Shot') +
par(pty='s') +
hvline(yintercept=22.5)
  scale_fill_continuous(name = 'Average Points per shot')

#still have to change legend titles!!

detach(shot)
```

###############
#Train test split
###############

```
#train proportion
train_sample = 0.5
smp_size <- floor(train_sample * nrow(join2))

## set the seed to make your partition reproducible
set.seed(6)
train_ind <- sample(seq_len(nrow(join2)), size = smp_size)

nba_train <- join2[train_ind, ]
nba_test <- join2[-train_ind, ]
```

###################
#Modeling - Logistics
###################

```
### This will turn off warning messages
options(warn=-1)
###Drop useless variables: e.g.unique id or name for each player
nba_train_sub <- subset(nba_train,select =
-c(GAME_ID,MATCHUP,SHOT_RESULT,CLOSEST_DEFENDER,CLOSEST_DEFENDER_
PLAYER_ID,

PTS,player_name,player_id,country,SHOT_CLOCK_24,SHOT_CLOCK_NA))
```

```
nba_test_sub <- subset(nba_test,select =
-c(GAME_ID,MATCHUP,SHOT_RESULT,CLOSEST_DEFENDER,CLOSEST_DEFENDER_
PLAYER_ID,

PTS,player_name,player_id,country,SHOT_CLOCK_24,SHOT_CLOCK_NA))

####10-fold cross validation for Logistic and Null Model
n <- nrow(nba_train)
nfold <- 10
OOS <- data.frame(logistic=NA,null=NA)
foldid <- rep(1:nfold,each=ceiling(n/nfold))[sample(1:n)]
### create an empty dataframe of results
OOS <- data.frame(logistic=rep(NA,nfold), null=rep(NA,nfold))
### Use a for loop to run through the nfold trails
for(k in 1:nfold){
  train <- which(foldid!=k) # train on all but fold `k'
  ## fit the two regressions and null model
  model.logistic <-glm(FGM~., data=nba_train_sub, subset=train,family="binomial")
  model.nulll <-glm(FGM~1, data=nba_train_sub, subset=train,family="binomial")
  ## get predictions: type=response so we have probabilities
  pred.logistic             <- predict(model.logistic, newdata=nba_train_sub[-train,],
type="response")
  pred.null <- predict(model.nulll, newdata=nba_train_sub[-train,], type="response")

  ## calculate and log R2
  # Logistic
  OOS$logistic[k] <- R2(y=nba_train_sub$FGM[-train], pred=pred.logistic, family="binomial")
  OOS$logistic[k]
  #Null
  OOS$null[k] <- R2(y=nba_train_sub$FGM[-train], pred=pred.null, family="binomial")
  OOS$null[k]
  #Null Model guess
  sum(nba_train_sub$FGM[train]==1)/length(train)

  ## We will loop this nfold times (I setup for 10)
  ## this will print the progress (iteration that finished)
  print(paste("Iteration",k,"of",nfold,"(thank you for your patience)"))
}
###
### List the mean of the results stored in the dataframe OOS
### we have nfold values in OOS for each model, this computes the mean of them)
colMeans(OOS)
m.OOS <- as.matrix(OOS)
rownames(m.OOS) <- c(1:nfold)
barplot(t(as.matrix(OOS)), beside=TRUE, legend=TRUE, args.legend=c(xjust=1, yjust=0.5),
        ylab= bquote( "Out of Sample " ~ R^2), xlab="Fold", names.arg = c(1:10))
```

```
############################################
### Modeling: Lasso
#### install package for Lasso
installpkg("glmnet")
library(glmnet)
#### Run Lasso
#### the features need to be a matrix ([,-1] removes the first column which is the intercept)
Mx<- model.matrix(FGM~.^2, data=nba_train_sub)[,-1]
My<- nba_train_sub$FGM == 1
### This defined the features we will use the matrix Mx (X) and the target My (Y)
###
#### Lasso requires a penalty parameter lambda
num.features <- ncol(Mx)
num.n <- nrow(Mx)
num.FGM <- sum(My)
w <- (num.FGM/num.n)*(1-(num.FGM/num.n))
#### For the binomial case, a theoretically valid choice is
lambda.theory <- sqrt(w*log(num.features/0.05)/num.n)
###
### next we call Lasso providing the
### features as matrix Mx
### the target as a vector My
### telling it is for logistic, family="binomial"
### and specifying lambda = lambda.theory
lassoTheory <- glmnet(Mx,My, family="binomial",lambda = lambda.theory)
### by calling the summary we see the list of object in sclassoTheory
summary(lassoTheory)
support(lassoTheory$beta)
### these are the labels
colnames(Mx)[support(lassoTheory$beta)]
### there are in total
length(support(lassoTheory$beta))
### coefficients selected by the model using the theoretical choice
###
###
### If we omit the lambda, the function will solve for all values of lambda
lasso <- glmnet(Mx,My, family="binomial")
### By running for all vaules of lambda we get many models back
### now we have the summary
summary(lasso)

### Compute the Lasso for all values of lambda,
### the whole "path" of models can be evaluated by a OOS experiment
### the following command yields a cross validation procedure
### the following command takes some time.
```

```
lassoCV <- cv.glmnet(Mx,My, family="binomial")
### Plot the fitting graph
### red dots are mean values and the bars are the uncertainty
par(mar=c(1.5,1.5,2,1.5))
par(mai=c(1.5,1.5,2,1.5))
plot(lassoCV, main="Fitting Graph for CV Lasso \n \n # of non-zero coefficients  ", xlab =
expression(paste("log(",lambda,")")))
par(pty = "m")
### Compute lambda.min
lassoCV$lambda[which.min(lassoCV$cvm)]
### sclassoCV$cvm has the mean values
### which.min(sclassoCV$cvm) returns the index that minimizes it
### in any case we have lambda.min and lambda.1se.
###
### lambda.min is perceived as aggressive (picks too many variables)
### lambda.1se is perceived as conservative (picks too few variables)
###
### we plot them in the previous picture
text(log(lassoCV$lambda.min), .95,"min",cex=1)
text(log(lassoCV$lambda.1se), 1,"1se",cex=1)
lines(c(log(lambda.theory),log(lambda.theory)),c(0.3,2.4),lty=3,col="blue")
text(log(lambda.theory), 1.05,"theory",cex=1)
length(support(lasso$beta[,which.min(lassoCV$cvm)]))


#### Post Lasso #####
#### we sepect a model based on the proposed rules
#### and perform cross validation
### The code is the same as the CV before
PL.OOS <- data.frame(PL.min=rep(NA,nfold), PL.1se=rep(NA,nfold),
PL.theory=rep(NA,nfold))
L.OOS <- data.frame(L.min=rep(NA,nfold), L.1se=rep(NA,nfold), L.theory=rep(NA,nfold))
features.min <- support(lasso$beta[,which.min(lassoCV$cvm)])
colnames(Mx)[features.min]
length(features.min)
features.1se <- support(lasso$beta[,which.min( (lassoCV$lambda-lassoCV$lambda.1se)^2)])
length(features.1se)
features.theory <- support(lassoTheory$beta)
length(features.theory)
data.min <- data.frame(Mx[,features.min],My)
nrow(Mx[,features.min])
data.1se <- data.frame(Mx[,features.1se],My)
data.theory <- data.frame(Mx[,features.theory],My)
for(k in 1:nfold){
  train <- which(foldid!=k) # train on all but fold `k'
  ### This is the CV for the Post Lasso Estimates
  rmin <- glm(My~., data=data.min, subset=train, family="binomial")
  if ( length(features.1se) == 0){  r1se <- glm(FGM~1, data=nba_train_sub, subset=train,
family="binomial")
  } else {r1se <- glm(My~., data=data.1se, subset=train, family="binomial")
  }

  if ( length(features.theory) == 0){
          rtheory <- glm(FGM~1, data=nba_train_sub, subset=train, family="binomial")
  } else {rtheory <- glm(My~., data=data.theory, subset=train, family="binomial") }


  predmin <- predict(rmin, newdata=data.min[-train,], type="response")
  pred1se  <- predict(r1se, newdata=data.1se[-train,], type="response")
  predtheory <- predict(rtheory, newdata=data.theory[-train,], type="response")
  PL.OOS$PL.min[k] <- R2(y=My[-train], pred=predmin, family="binomial")
  PL.OOS$PL.1se[k] <- R2(y=My[-train], pred=pred1se, family="binomial")
  PL.OOS$PL.theory[k] <- R2(y=My[-train], pred=predtheory, family="binomial")

  ### This is the CV for the Lasso estimates
  lassomin  <- glmnet(Mx[train,],My[train], family="binomial",lambda = lassoCV$lambda.min)
  lasso1se  <- glmnet(Mx[train,],My[train], family="binomial",lambda = lassoCV$lambda.1se)
  lassoTheory <- glmnet(Mx[train,],My[train], family="binomial",lambda = lambda.theory)

  predlassomin <- predict(lassomin, newx=Mx[-train,], type="response")
  predlasso1se  <- predict(lasso1se, newx=Mx[-train,], type="response")
  predlassotheory <- predict(lassoTheory, newx=Mx[-train,], type="response")
  L.OOS$L.min[k] <- R2(y=My[-train], pred=predlassomin, family="binomial")
  L.OOS$L.1se[k] <- R2(y=My[-train], pred=predlasso1se, family="binomial")
  L.OOS$L.theory[k] <- R2(y=My[-train], pred=predlassotheory, family="binomial")

  print(paste("Iteration",k,"of",nfold,"completed"))
}

R2performance <- cbind(PL.OOS,L.OOS,OOS)
colMeans(R2performance)
par( mar=  c(8, 4, 4, 2) + 0.6 )
barplot(colMeans(R2performance), las=2,xpd=FALSE, ylim=c(0,.3) , xlab="", ylab = bquote(
"Average Out of Sample " ~ R^2))
m.OOS <- as.matrix(R2performance)
rownames(m.OOS) <- c(1:nfold)
par(mar=c(1.5,1.5,1.5,3))
par(mai=c(1.5,1.5,1.5,3))
barplot(t(as.matrix(m.OOS)), beside=TRUE, ylim=c(0,.4) ,legend=TRUE, args.legend=c(x=
"topright", y=0.92,bty = "n"),
        ylab= bquote( "Out of Sample " ~ R^2), xlab="Fold", names.arg = c(1:10))
```

```
### We do the same for
### accuracy Performance
PL.OOS.TP <- data.frame(PL.min=rep(NA,nfold), PL.1se=rep(NA,nfold),
PL.theory=rep(NA,nfold))
L.OOS.TP <- data.frame(L.min=rep(NA,nfold), L.1se=rep(NA,nfold), L.theory=rep(NA,nfold))
PL.OOS.TN <- data.frame(PL.min=rep(NA,nfold), PL.1se=rep(NA,nfold),
PL.theory=rep(NA,nfold))
L.OOS.TN <- data.frame(L.min=rep(NA,nfold), L.1se=rep(NA,nfold), L.theory=rep(NA,nfold))
PL.OOS.FP <- data.frame(PL.min=rep(NA,nfold), PL.1se=rep(NA,nfold),
PL.theory=rep(NA,nfold))
L.OOS.FP <- data.frame(L.min=rep(NA,nfold), L.1se=rep(NA,nfold), L.theory=rep(NA,nfold))
PL.OOS.FN <- data.frame(PL.min=rep(NA,nfold), PL.1se=rep(NA,nfold),
PL.theory=rep(NA,nfold))
L.OOS.FN <- data.frame(L.min=rep(NA,nfold), L.1se=rep(NA,nfold), L.theory=rep(NA,nfold))
OOS.TP <- data.frame(logistic=rep(NA,nfold),null=rep(NA,nfold))
OOS.TN <- data.frame(logistic=rep(NA,nfold),null=rep(NA,nfold))
OOS.FP <- data.frame(logistic=rep(NA,nfold),null=rep(NA,nfold))
OOS.FN <- data.frame(logistic=rep(NA,nfold),null=rep(NA,nfold))
val <- .5
for(k in 1:nfold){
  train <- which(foldid!=k) # train on all but fold `k'

  ### This is the CV for the Post Lasso Estimates
  rmin <- glm(My~., data=data.min, subset=train, family="binomial")
  if ( length(features.1se) == 0){  r1se <- glm(FGM~1, data=nba_train_sub, subset=train,
family="binomial")
  } else {r1se <- glm(My~., data=data.1se, subset=train, family="binomial")
  }

  if ( length(features.theory) == 0){
           rtheory <- glm(FGM~1, data=nba_train_sub, subset=train, family="binomial")
  } else {rtheory <- glm(My~., data=data.theory, subset=train, family="binomial") }


  predmin <- predict(rmin, newdata=data.min[-train,], type="response")
  pred1se  <- predict(r1se, newdata=data.1se[-train,], type="response")
  predtheory <- predict(rtheory, newdata=data.theory[-train,], type="response")

  values <- FPR_TPR( (predmin >= val) , My[-train] )
  PL.OOS$PL.min[k] <- values$ACC
  PL.OOS.TP$PL.min[k] <- values$TP
  PL.OOS.TN$PL.min[k] <- values$TN
  PL.OOS.FP$PL.min[k] <- values$FP
  PL.OOS.FN$PL.min[k] <- values$FN

  values <- FPR_TPR( (pred1se >= val) , My[-train] )
  PL.OOS$PL.1se[k] <- values$ACC
  PL.OOS.TP$PL.1se[k] <- values$TP
  PL.OOS.FP$PL.1se[k] <- values$FP
  PL.OOS.TN$PL.1se[k] <- values$TN
  PL.OOS.FN$PL.1se[k] <- values$FN
  values <- FPR_TPR( (predtheory >= val) , My[-train] )
  PL.OOS$PL.theory[k] <- values$ACC

  ### This is the CV for the Lasso estimates
  lassomin  <- glmnet(Mx[train,],My[train], family="binomial",lambda = lassoCV$lambda.min)
  lasso1se  <- glmnet(Mx[train,],My[train], family="binomial",lambda = lassoCV$lambda.1se)
  lassoTheory <- glmnet(Mx[train,],My[train], family="binomial",lambda = lambda.theory)

  predlassomin <- predict(lassomin, newx=Mx[-train,], type="response")
  predlasso1se  <- predict(lasso1se, newx=Mx[-train,], type="response")
  predlassotheory <- predict(lassoTheory, newx=Mx[-train,], type="response")
  values <- FPR_TPR( (predlassomin >= val) , My[-train] )
  L.OOS$L.min[k] <- values$ACC
  L.OOS.TP$L.min[k] <- values$TP
  L.OOS.TN$L.min[k] <- values$TN
  L.OOS.FP$L.min[k] <- values$FP
  L.OOS.FN$L.min[k] <- values$FN
  values <- FPR_TPR( (predlasso1se >= val) , My[-train] )
  L.OOS$L.1se[k] <- values$ACC
  L.OOS.TP$L.1se[k] <- values$TP
  L.OOS.TN$L.1se[k] <- values$TN
  L.OOS.FP$L.1se[k] <- values$FP
  L.OOS.FN$L.1se[k] <- values$FN
  values <- FPR_TPR( (predlassotheory >= val) , My[-train] )
  L.OOS$L.theory[k] <- values$ACC
  L.OOS.TP$L.theory[k] <- values$TP
  L.OOS.TN$L.theory[k] <- values$TN
  L.OOS.FP$L.theory[k] <- values$FP
  L.OOS.FN$L.theory[k] <- values$FN


  ## fit the two regressions and null model
  ##### full model uses all 200 signals
  model.logistic <-glm(FGM~., data=nba_train_sub, subset=train,family="binomial")
  model.nulll <-glm(FGM~1, data=nba_train_sub, subset=train,family="binomial")
  ## get predictions: type=response so we have probabilities
  pred.logistic             <- predict(model.logistic, newdata=nba_train_sub[-train,],
type="response")
  pred.null <- predict(model.nulll, newdata=nba_train_sub[-train,], type="response")

  ## calculate and log R2
  # Logistic
  values <- FPR_TPR( (pred.logistic >= val) , My[-train] )
  OOS$logistic[k] <- values$ACC
  OOS.TP$logistic[k] <- values$TP
  OOS.TN$logistic[k] <- values$TN
  OOS.FP$logistic[k] <- values$FP
  OOS.FN$logistic[k] <- values$FN
  #Null
  values <- FPR_TPR( (pred.null >= val) , My[-train] )
  OOS$null[k] <- values$ACC
  OOS.TP$null[k] <- values$TP
  OOS.TN$null[k] <- values$TN
  OOS.FP$null[k] <- values$FP
  OOS.FN$null[k] <- values$FN

  print(paste("Iteration",k,"of",nfold,"completed"))
}
par(mar=c(1,1,1,1))
par(mai=c(1,1,1,1))
ACCperformance <- cbind(PL.OOS,L.OOS,OOS)
colMeans(ACCperformance)
barplot(colMeans(ACCperformance), xpd=FALSE, ylim=c(.5,.66), xlab="Method", ylab =
"Accuracy")
m.OOS <- as.matrix(ACCperformance)
rownames(m.OOS) <- c(1:nfold)
par(mar=c(1.5,1.5,1.5,1))
par(mai=c(1.5,1.5,1.5,1))
barplot(t(as.matrix(m.OOS)), beside=TRUE,xlim = c(0.8,98),ylim = c(0,0.7),legend=TRUE,
args.legend=c(xjust=0.4, yjust=0.5),
        ylab= bquote( "Out of Sample Accuracy"), xlab="Fold", names.arg = c(1:10))

#########
####GET ROC curve
installpkg("pROC")
library(pROC)
### Post Lasso with min lambda- prediction on test sample
rmin <- glm(My~., data=data.min, family="binomial")
Mx_plmin<- model.matrix(FGM~.^2, data=nba_test_sub)[,-1]
My_plmin<- nba_test_sub$FGM
data.min_pl <- data.frame(Mx_plmin[,features.min],My_plmin)
predmin_pl <- predict(rmin, newdata=data.min_pl, type="response")
roc_min_pl <- roc(nba_test_sub$FGM,predmin_pl,legacy.axes =TRUE,xlab = "False Positive
Rate",ylab = "True Positive Rate")
roc_min_pl
```

```
plot.roc(nba_test_sub$FGM,predmin_pl,legacy.axes =TRUE,xlab = "False Positive
Rate",ylab = "True Positive Rate",col = "#377eb8",lwd =2.5,print.auc = TRUE,print.auc.x =
0.5)
legend("bottomright",legend = c("PL.min Model"),col = c("#377eb8"),lwd = 2.5)

#########################
#Modeling: Random Forest
##########################
#Random Forest

nba_trf = subset(nba_train, select = -c(GAME_ID, MATCHUP, SHOT_RESULT,
CLOSEST_DEFENDER, CLOSEST_DEFENDER_PLAYER_ID, PTS, player_name,
player_id, country, SHOT_CLOCK_24, SHOT_CLOCK_NA))
#head(nba_trf)
nba_trf <- nba_trf[c('FGM', setdiff(names(nba_trf), 'FGM'))]

x <- nba_trf[,2:29]
y <- nba_trf[,1]

#simple model first

j1 <- randomForest(
  formula = as.factor(FGM) ~ .,
  data   = nba_trf
)

#plot results to check the optimal number of trees
plot(j1)
layout(matrix(c(1,2),nrow=1),
         width=c(4,1))
par(mar=c(5,4,4,0)) #No margin on the right side
plot(j1, log="y")
par(mar=c(5,0,4,2)) #No margin on the left side
plot(c(0,1),type="n", axes=F, xlab="", ylab="")
legend("top", colnames(j1$err.rate),col=1:4,cex=0.8,fill=1:4)

#####################
#Tuning

# names of features
features <- setdiff(names(nba_trf), "FGM")

set.seed(22)

j2 <- tuneRF(
  x       = nba_trf[features],
  y        = as.factor(nba_trf$FGM),
  ntreeTry   = 200,
  mtryStart  = 7,
  stepFactor = 1.5,
  improve        = 0.01,
  trace            = FALSE         # to not show real-time progress
)


# hyperparameter grid search
hyper_grid <- expand.grid(
  mtry             = seq(3, 13, by = 2),
  node_size  = seq(3, 9, by = 2),
  sampe_size = c(.55, .632, .70, .80),
  OOB_RMSE   = 0
)

# total number of combinations
nrow(hyper_grid)
## [1] 96

library(ranger)

for(i in 1:nrow(hyper_grid)) {

  # train model
  model <- ranger(
          formula          = as.factor(FGM) ~ .,
          data               = nba_trf,
          num.trees       = 200,
          mtry               = hyper_grid$mtry[i],
          min.node.size   = hyper_grid$node_size[i],
          sample.fraction = hyper_grid$sampe_size[i],
          seed               = 22
  )

  # add OOB error to grid
  hyper_grid$OOB_RMSE[i] <- sqrt(model$prediction.error)
}

hyper_grid %>%
  dplyr::arrange(OOB_RMSE) %>%
  head(10)
```

```
j4 <- randomForest(as.factor(FGM)~., data=nba_trf, nodesize=9, mtry=5, sample.fraction
=0.55, ntree =200)

j5 <- randomForest(as.factor(FGM)~., data=nba_trf, nodesize=9, mtry=3, ntree =200)

library(pROC)
require(pROC)
rf.roc<-roc(nba_trf$FGM,j4$votes[,2],legacy.axes =TRUE,xlab = "False Positive Rate",ylab =
"True Positive Rate",col = "#377eb8",lwd = 2.5,print.auc = TRUE)
plot.roc(nba_trf$FGM,j4$votes[,2],legacy.axes =TRUE,xlab = "False Positive Rate",ylab =
"True Positive Rate",col = "#377eb8",lwd = 2.5,print.auc = TRUE)
legend("bottomright",legend = c("Random Forest"),col = c("#377eb8"),lwd = 2.5)
par(pty = "s")
auc(rf.roc)

importance(j4)
varImp(j4)
varImpPlot(j4)

varImpPlot(j4, sort=TRUE, n.var=min(15, nrow(j4$importance)), type=NULL, class=NULL,
scale=TRUE, main = 'Variable Importance')

################################
#test the results


nba_test_rf = subset(nba_test, select = -c(GAME_ID, MATCHUP, SHOT_RESULT,
CLOSEST_DEFENDER, CLOSEST_DEFENDER_PLAYER_ID, PTS, player_name,
player_id, country, SHOT_CLOCK_24, SHOT_CLOCK_NA))
#head(nba_trf)
nba_test_trf <- nba_test[c('FGM', setdiff(names(nba_test_rf), 'FGM'))]

test <-predict(j4, newdata = nba_test_trf, type='prob')

#paste results of the model
nba_test <- cbind(nba_test, test)

#calculate expected points
nba_test$expected_points <- nba_test$'1' * nba_test$PTS_TYPE

#points over expectation
nba_test$points_diff <- nba_test$PTS - nba_test$expected_points



#aggregate per player
attach(nba_test)
```

```
playerdata <-aggregate(nba_test, by=list(player_name, nba_test$cluster),
                       FUN=mean)
num_shots <-nba_test %>% count(player_name, cluster)
app <-left_join(playerdata, num_shots,
                       by = c('player_name', 'cluster'))

app2<- app %>% filter(n > 30)

write.csv(playerdata, file = 'playerdata.csv')
write.csv(num_shots, file = 'shot_num.csv')


player_app <- ggplot(app, aes(points_diff, expected_points))
player_app + geom_point(aes(size=n, color=), shape=1)+
  scale_colour_gradientn(colours = terrain.colors(10)) +
  xlab('Points above expected') +
  ylab('Expected points') +
  ggtitle('Model prediction by player') +
  scale_fill_continuous(name = 'Average Points per shot')

salary_pae <- ggplot(app, aes(points_diff, salary))
salary_pae + geom_point(aes(size=n), shape=1)+
  scale_colour_gradientn(colours = terrain.colors(10)) +
  xlab('Points Above Expectation') +
  ylab('Salary') +
  ggtitle('Salary and model prediction by player') +
  geom_smooth(method='lm', se = FALSE) +
  scale_fill_continuous(name = 'Average Points per shot')


detach(nba)



roc(nba_test, nba_test$'1')
#, plot=TRUE, legacy.axes=TRUE, percent=TRUE, xlab="False Positive Percentage",
ylab="True Postive Percentage", col="#377eb8", lwd=4, print.auc=TRUE)


result.roc <- roc(nba_test$FGM, nba_test$'1') # Draw ROC curve.
result.roc
plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")

result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold",
"accuracy"))
print(result.coords)#to get threshold and accuracy
```

```
#######
#Cluster
#######

installpkg("ElemStatLearn")
library(ElemStatLearn)
installpkg("class")
library(class)
names(join2)
xjoin2 <- scale(join2[,c(3:12,17,27:38)])
summary(xjoin2)
apply(xjoin2,2,sd) # sd=1
apply(xjoin2,2,mean) # mean=0
View(xjoin2)
## First we fit k-Means with k=2 (i.e. two clusters)
two <- kmeans(xjoin2,2,nstart=10)
kfit <- lapply(1:100, function(k) kmeans(xjoin2,k))
kaic <- sapply(kfit,kIC)
kbic  <- sapply(kfit,kIC,"B")
kHDic  <- sapply(kfit,kIC,"C")
## Now we plot them, first we plot AIC
par(mar=c(1,1,1,1))
par(mai=c(1,1,1,1))
plot(kaic, xlab="k (# of clusters)", ylab="IC (Deviance + Penalty)",
    ylim=range(c(kaic,kbic,kHDic)), # get them on same page
    type="l", lwd=2)
abline(v=which.min(kaic))
lines(kbic, col=4, lwd=2)
abline(v=which.min(kbic),col=4)
lines(kHDic, col=3, lwd=2)
abline(v=which.min(kHDic),col=3)
text(c(which.min(kaic),which.min(kbic),which.min(kHDic)),c(mean(kaic),mean(kbic),mean(kH
Dic)),c("AIC","BIC","HDIC"))
which.min(kaic)
which.min(kbic)
which.min(kHDic)
k=100#AIC
k=100#BIC
k=16#HDIC
k=5
# BIC: 56%; HDIC: 40%; k=5 26% (for explaining deviance of x)
1 - sum(kfit[[k]]$tot.withinss)/kfit[[k]]$totss


Ssimple_kmeans <- kmeans(xjoin2,10,nstart=10)
colorcluster <- 1+Ssimple_kmeans$cluster
```

```
### Summarize a variable on each cluster
a0=tapply(join2[,9],Ssimple_kmeans$cluster,mean)
a1=tapply(join2[,10],Ssimple_kmeans$cluster,mean)
a2=tapply(join2[,11],Ssimple_kmeans$cluster,mean)
a3=tapply(join2[,12],Ssimple_kmeans$cluster,mean)
a4=tapply(join2[,13],Ssimple_kmeans$cluster,mean)
a5=tapply(join2[,17],Ssimple_kmeans$cluster,mean)
a6=tapply(join2[,24],Ssimple_kmeans$cluster,mean)
a7=tapply(join2[,25],Ssimple_kmeans$cluster,mean)
a8=tapply(join2[,26],Ssimple_kmeans$cluster,mean)
a9=tapply(join2[,33],Ssimple_kmeans$cluster,mean)
a10=tapply(join2[,34],Ssimple_kmeans$cluster,mean)
a11=tapply(join2[,35],Ssimple_kmeans$cluster,mean)
a12=tapply(join2[,36],Ssimple_kmeans$cluster,mean)
a13=tapply(join2[,38],Ssimple_kmeans$cluster,mean)
df=data.frame(cbind(a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13))
View(df)
getwd()
### We can compute the wins on each cluster
### and contrast with the overall FGM
tapply(join2[,18],Ssimple_kmeans$cluster,mean,na.rm=TRUE)
mean(join2[,18],na.rm=TRUE)


###############################
#plot cluster plot
names(join2)
simple <- join2[,c(9,12)]
par(mar = c(5, 5, 3, 1))
Ssimple <- scale(simple)
apply(Ssimple,2,sd)
apply(Ssimple,2,mean)
Ssimple_kmeans1 <- kmeans(Ssimple,10,nstart=10)
colorcluster <- 1+Ssimple_kmeans1$cluster

plot(Ssimple, col = 1, xlab="SHOT_CLOCK",
   ylab="SHOT_DIST",ylim=c(-1.5,2), cex = 0.5,
   main="k Means for shot distance and shot clock")
plot(Ssimple, col = colorcluster, xlab="SHOT_CLOCK",
   ylab="SHOT_DIST",ylim=c(-1.5,2), cex = 0.5,
   main="k Means for shot distance and shot clock")
points(Ssimple_kmeans1$centers, col = 1, pch = 24,
    cex = 1.5, lwd=1, bg = 2:5,
    main="k Means for shot distance and shot clock")

### The command
```

```
Ssimple_kmeans$centers
## displays the k centers (good to interpret)
Ssimple_kmeans$size
## displays the size of each cluster
Ssimple_kmeans$cluster
b4=as.vector(Ssimple_kmeans$cluster)
names(join2)
join3=cbind(join2[,1],join2[,6],join2[,20],b4)
View(join3)
getwd()
write.csv(join3,'C:/Users/Cara You/Desktop/duke/data science for
business/project/Cara.csv', row.names = FALSE)
b1=Ssimple_kmeans$size
b2=Ssimple_kmeans$size/127745
b3=tapply(join2[,18],Ssimple_kmeans$cluster,mean,na.rm=TRUE)
mean(join2[,18],na.rm=TRUE)
df=data.frame(cbind(b1,b2,b3))
View(df)


##########################
# ATTACH CLUSTER RESULTS
##########################
cluster <- read.csv('clstr10.csv')

join3 <- cbind(join2, cluster$CLUSTER)

nba_test <- join3[-train_ind, ]

colnames(nba_test)[colnames(nba_test)=='cluster$CLUSTER'] <- 'cluster'
```