# NBAAnalysisConferenceFinalsClassificationModel.R

dpesl

2020-05-17

```r
##Import NBA Salary Data
NBASalaryAnalysisData <- read.csv("C:/Users/dpesl/Desktop/NBASalaryAnalysisData.csv",
                                  header = TRUE)
NBASalaryAnalysisData2020 <- read.csv("C:/Users/dpesl/Desktop/NBASalaryAnalysisData2020.csv",
                                      header = TRUE)

##Remove first column (row numbers)
NBASalaryAnalysisData <- NBASalaryAnalysisData[,-1]
NBASalaryAnalysisData2020 <- NBASalaryAnalysisData2020[,-1]

##install.packages("matrixStats")
library(matrixStats)
```

```
## Warning: package 'matrixStats' was built under R version 3.6.3
```

```r
##Let us separate perGame and perPoss metrics
MasterPerGame <- NBASalaryAnalysisData[,-(78:121)]
MasterPerGame2020 <- NBASalaryAnalysisData2020[,-(76:119)]
MasterPerGame[,9] <- as.character(MasterPerGame[,9])
for (i in 1:dim(MasterPerGame)[1]) {
  if(MasterPerGame[i,9] == 'CHAMPIONS'){
    MasterPerGame[i,9] <- 0
  }
  if(MasterPerGame[i,9] == 'FINALS'){
    MasterPerGame[i,9] <- 1
  }
  if(MasterPerGame[i,9] == 'CFINALS'){
    MasterPerGame[i,9] <- 2
  }
  if(MasterPerGame[i,9] == '2R'){
    MasterPerGame[i,9] <- 3
  }
  if(MasterPerGame[i,9] == '1R'){
    MasterPerGame[i,9] <- 4
  }
  if(MasterPerGame[i,9] == 'MISSED'){
    MasterPerGame[i,9] <- 5
  }
}
MasterPerGame[,9] <- as.numeric(MasterPerGame[,9])
##Note that we scale variables according to season
```

```r
##this is done because we want to avoid running into problems with
##changes in game plans (we will see whether teams are better at 3pts compared
##to league in a paricular season, vs over 29 seasons)
##then we re-scale all together
MasterPerGame2020[,-c((1:5),8)] <- scale(MasterPerGame2020[,-c((1:5),8)])
MasterPerGame[(1:27),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(1:27),-c((1:5),7,(9:10))])
MasterPerGame[(28:54),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(28:54),-c((1:5),7,(9:10))])
MasterPerGame[(55:81),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(55:81),-c((1:5),7,(9:10))])
MasterPerGame[(82:108),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(82:108),-c((1:5),7,(9:10))])
MasterPerGame[(109:135),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(109:135),-c((1:5),7,(9:10))])
MasterPerGame[(136:164),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(136:164),-c((1:5),7,(9:10))])
MasterPerGame[(165:193),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(165:193),-c((1:5),7,(9:10))])
MasterPerGame[(194:222),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(194:222),-c((1:5),7,(9:10))])
MasterPerGame[(223:251),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(223:251),-c((1:5),7,(9:10))])
MasterPerGame[(252:280),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(252:280),-c((1:5),7,(9:10))])
MasterPerGame[(281:309),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(281:309),-c((1:5),7,(9:10))])
MasterPerGame[(310:338),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(310:338),-c((1:5),7,(9:10))])
MasterPerGame[(339:367),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(339:367),-c((1:5),7,(9:10))])
MasterPerGame[(368:396),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(368:396),-c((1:5),7,(9:10))])
MasterPerGame[(397:426),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(397:426),-c((1:5),7,(9:10))])
MasterPerGame[(427:456),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(427:456),-c((1:5),7,(9:10))])
MasterPerGame[(457:486),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(457:486),-c((1:5),7,(9:10))])
MasterPerGame[(487:516),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(487:516),-c((1:5),7,(9:10))])
MasterPerGame[(517:546),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(517:546),-c((1:5),7,(9:10))])
MasterPerGame[(547:576),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(547:576),-c((1:5),7,(9:10))])
MasterPerGame[(577:606),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(577:606),-c((1:5),7,(9:10))])
MasterPerGame[(607:636),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(607:636),-c((1:5),7,(9:10))])
MasterPerGame[(637:666),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(637:666),-c((1:5),7,(9:10))])
MasterPerGame[(667:696),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(667:696),-c((1:5),7,(9:10))])
MasterPerGame[(697:726),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(697:726),-c((1:5),7,(9:10))])
MasterPerGame[(727:756),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(727:756),-c((1:5),7,(9:10))])
MasterPerGame[(757:786),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(757:786),-c((1:5),7,(9:10))])
MasterPerGame[(787:816),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(787:816),-c((1:5),7,(9:10))])
MasterPerGame[(817:846),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(817:846),-c((1:5),7,(9:10))])
MasterPerGame2020[,-c((1:5),8)] <- (MasterPerGame2020[,-c((1:5),8)] - colMeans(MasterPerGame[,-c((1:5),
MasterPerGame[,-c((1:5),7,(9:10))] <- scale(MasterPerGame[,-c((1:5),7,(9:10))])
MasterPerGame <- MasterPerGame[,-c((1:5),7,10,(12:14),19,20,34,35)]
MasterPerGame2020 <- MasterPerGame2020[,-c((1:5),8,(10:12),17,18,32,33)]
MasterPerPoss <- NBASalaryAnalysisData[,-(34:77)]
MasterPerPoss[,9] <- as.character(MasterPerPoss[,9])
for (i in 1:dim(MasterPerPoss)[1]) {
  if(MasterPerPoss[i,9] == 'CHAMPIONS'){
    MasterPerPoss[i,9] <- 0
  }
  if(MasterPerPoss[i,9] == 'FINALS'){
    MasterPerPoss[i,9] <- 1
  }
  if(MasterPerPoss[i,9] == 'CFINALS'){
    MasterPerPoss[i,9] <- 2
  }
  if(MasterPerPoss[i,9] == '2R'){
    MasterPerPoss[i,9] <- 3
  }
  if(MasterPerPoss[i,9] == '1R'){
    MasterPerPoss[i,9] <- 4
  }
  if(MasterPerPoss[i,9] == 'MISSED'){
    MasterPerPoss[i,9] <- 5
  }
}
MasterPerPoss[,9] <- as.numeric(MasterPerPoss[,9])
MasterPerPoss[(1:27),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(1:27),-c((1:5),7,(9:10))])
MasterPerPoss[(28:54),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(28:54),-c((1:5),7,(9:10))])
MasterPerPoss[(55:81),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(55:81),-c((1:5),7,(9:10))])
MasterPerPoss[(82:108),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(82:108),-c((1:5),7,(9:10))])
MasterPerPoss[(109:135),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(109:135),-c((1:5),7,(9:10))])
MasterPerPoss[(136:164),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(136:164),-c((1:5),7,(9:10))])
MasterPerPoss[(165:193),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(165:193),-c((1:5),7,(9:10))])
MasterPerPoss[(194:222),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(194:222),-c((1:5),7,(9:10))])
MasterPerPoss[(223:251),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(223:251),-c((1:5),7,(9:10))])
MasterPerPoss[(252:280),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(252:280),-c((1:5),7,(9:10))])
MasterPerPoss[(281:309),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(281:309),-c((1:5),7,(9:10))])
MasterPerPoss[(310:338),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(310:338),-c((1:5),7,(9:10))])
MasterPerPoss[(339:367),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(339:367),-c((1:5),7,(9:10))])
MasterPerPoss[(368:396),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(368:396),-c((1:5),7,(9:10))])
MasterPerPoss[(397:426),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(397:426),-c((1:5),7,(9:10))])
MasterPerPoss[(427:456),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(427:456),-c((1:5),7,(9:10))])
MasterPerPoss[(457:486),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(457:486),-c((1:5),7,(9:10))])
MasterPerPoss[(487:516),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(487:516),-c((1:5),7,(9:10))])
MasterPerPoss[(517:546),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(517:546),-c((1:5),7,(9:10))])
MasterPerPoss[(547:576),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(547:576),-c((1:5),7,(9:10))])
MasterPerPoss[(577:606),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(577:606),-c((1:5),7,(9:10))])
MasterPerPoss[(607:636),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(607:636),-c((1:5),7,(9:10))])
MasterPerPoss[(637:666),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(637:666),-c((1:5),7,(9:10))])
MasterPerPoss[(667:696),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(667:696),-c((1:5),7,(9:10))])
MasterPerPoss[(697:726),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(697:726),-c((1:5),7,(9:10))])
MasterPerPoss[(727:756),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(727:756),-c((1:5),7,(9:10))])
MasterPerPoss[(757:786),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(757:786),-c((1:5),7,(9:10))])
MasterPerPoss[(787:816),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(787:816),-c((1:5),7,(9:10))])
MasterPerPoss[(817:846),-c((1:5),7,(9:10))] <- scale(MasterPerPoss[(817:846),-c((1:5),7,(9:10))])
MasterPerPoss[,-c((1:5),7,(9:10))] <- scale(MasterPerPoss[,-c((1:5),7,(9:10))])
MasterPerPoss <- MasterPerPoss[,-c((1:5),7,10,(12:14),19,20,34,35)]

set.seed(2)
samplesize <- floor(0.25 * nrow(MasterPerGame))
Fold1index <- sample(seq_len(nrow(MasterPerGame)), samplesize)
PerGameFold1 <- MasterPerGame[Fold1index,]
Fold2index <- sample(seq_len(nrow(MasterPerGame[-Fold1index,])), samplesize)
PerGameFold2 <- MasterPerGame[Fold2index,]
Fold3index <- sample(seq_len(nrow(MasterPerGame[-c(Fold1index,Fold2index),])), (nrow(MasterPerGame)-2*sa
PerGameFold3 <- MasterPerGame[Fold3index,]
Fold4index <- sample(seq_len(nrow(MasterPerGame[-c(Fold1index,Fold2index,Fold3index),])), (nrow(MasterPe
PerGameFold4 <- MasterPerGame[Fold4index,]


##install.packages("ggplot2")
```

```r
library(ggplot2)
##install.packages("MLmetrics")
library(MLmetrics)
```

```
## Warning: package 'MLmetrics' was built under R version 3.6.3

##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall
```

```r
##install.packages("pROC")
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.6.3

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
##install.packages("MASS")
library(MASS)
##install.packages("caret")
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:MLmetrics':
##
##     MAE, RMSE
```

```r
##Conference Finalist Feature Selection
ytrain <- ceiling((MasterPerGame$finish-2)/5)
xtrain <- MasterPerGame[,-3]
datatrain <- cbind(ytrain, xtrain)

##Generalized Linear Model Feature Selection
set.seed(2)
cntrl <- rfeControl(functions = lrFuncs, method = "cv", number = 4, repeats = 10)
model.glm <- rfe(datatrain[,(2:63)], as.factor(datatrain[,1]), rfeControl = cntrl, sizes = c(5:25), met
## Warning in rfe.default(datatrain[, (2:63)], as.factor(datatrain[, 1]),
## rfeControl = cntrl, : Metric 'ROC' is not created by the summary function;
## 'Accuracy' will be used instead

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
model.glm
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (4 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          5   0.9149 0.6349   0.005465 0.02156
##          6   0.9173 0.6481   0.009136 0.02816
##          7   0.9137 0.6286   0.010388 0.03864
##          8   0.9125 0.6229   0.013651 0.04542
##          9   0.9184 0.6534   0.004593 0.01505        *
##         10   0.9078 0.5999   0.006121 0.03949
##         11   0.9043 0.5840   0.007040 0.02813
##         12   0.9042 0.5893   0.013691 0.04091
##         13   0.9031 0.5849   0.015858 0.06321
##         14   0.8983 0.5615   0.017654 0.06774
##         15   0.8995 0.5609   0.013779 0.05565
##         16   0.8983 0.5545   0.013817 0.05211
##         17   0.8971 0.5470   0.011990 0.04655
##         18   0.8983 0.5547   0.014806 0.05351
##         19   0.8971 0.5442   0.011990 0.04513
##         20   0.8995 0.5605   0.007156 0.02378
##         21   0.8972 0.5503   0.009797 0.03199
##         22   0.8948 0.5376   0.009180 0.06063
##         23   0.8983 0.5527   0.006233 0.03170
##         24   0.8983 0.5560   0.006122 0.03574
##         25   0.8983 0.5517   0.006122 0.04925
##         62   0.8830 0.4986   0.019319 0.11242
##
## The top 5 variables (out of 9):
##    Ranking, fg3mPerGameTeam, pctFG3PerGameTeam, pctFG3PerGameOpponent, fgaPerGameTeam
```

```
model.glm$optVariables
```

```
## [1] "Ranking"               "fg3mPerGameTeam"       "pctFG3PerGameTeam"
## [4] "pctFG3PerGameOpponent" "fgaPerGameTeam"        "ratioFTtoFGAOpponent"
## [7] "pctFG2PerGameTeam"     "fg3aPerGameTeam"       "pctFG2PerGameOpponent"
```

```r
##Discriminant Analysis Feature Selection
##Linear Discriminant
set.seed(2)
cntrl <- rfeControl(functions = ldaFuncs, method = "cv", number = 4, repeats = 10)
model.lda <- rfe(datatrain[,(2:63)], as.factor(datatrain[,1]), rfeControl = cntrl, sizes = c(5:25))
model.lda
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (4 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          5   0.9042 0.5314    0.00916 0.03424        *
##          6   0.9019 0.5255    0.01374 0.04841
##          7   0.8960 0.4902    0.01415 0.05700
##          8   0.9019 0.5193    0.01962 0.09686
##          9   0.9007 0.5226    0.01950 0.10694
##         10   0.8995 0.5197    0.01585 0.08739
##         11   0.8959 0.5004    0.01837 0.10131
##         12   0.8983 0.5067    0.01936 0.10860
##         13   0.8971 0.5025    0.02290 0.12870
##         14   0.8971 0.5025    0.02290 0.12870
##         15   0.8948 0.4881    0.02224 0.12115
##         16   0.8971 0.5013    0.02290 0.13249
##         17   0.8971 0.4996    0.02322 0.13767
##         18   0.8959 0.4932    0.02176 0.12824
##         19   0.8888 0.4694    0.02017 0.11106
##         20   0.8877 0.4631    0.01592 0.08095
##         21   0.8912 0.4807    0.01668 0.08996
##         22   0.8877 0.4649    0.01973 0.07964
##         23   0.8900 0.4754    0.02225 0.09730
##         24   0.8853 0.4435    0.02226 0.10713
##         25   0.8853 0.4435    0.02226 0.10713
##         62   0.8676 0.3904    0.01014 0.05012
##
## The top 5 variables (out of 5):
##    Ranking, winsTeam, nrtgTeamMisc, marginVictoryTeam, pctEFGTeamOppMisc
```

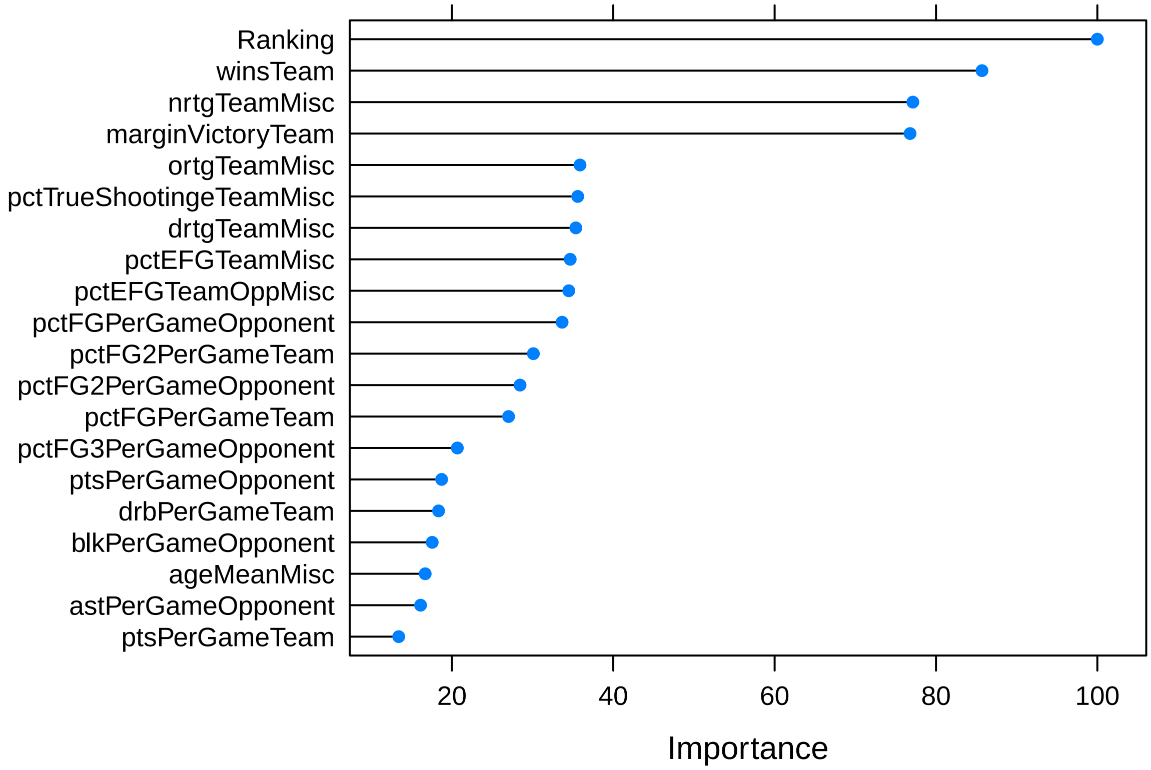
```r
model.lda$optVariables
```

```
## [1] "Ranking"           "winsTeam"          "nrtgTeamMisc"
## [4] "marginVictoryTeam" "pctEFGTeamOppMisc"
```

```r
##KNN Feature Selection
##Note we cannot apply rfe methods to KNN
##thus, we shall take variables with importance above 20%
model.knn <- train(as.factor(ytrain)~., data = datatrain,
                   trControl = trainControl(method = "cv", number = 4),
                   preProcess = c("center", "scale"), tuneGrid = expand.grid(k = seq(1,100, by = 1)),
                   method = "knn")
var.imp.knn <- varImp(model.knn)
print(var.imp.knn)
```

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 62)
##
##                           Overall
## Ranking                    100.00
## winsTeam                    85.70
## nrtgTeamMisc                77.13
## marginVictoryTeam           76.79
## ortgTeamMisc                35.88
## pctTrueShootingeTeamMisc    35.60
## drtgTeamMisc                35.36
## pctEFGTeamMisc              34.67
## pctEFGTeamOppMisc           34.49
## pctFGPerGameOpponent        33.66
## pctFG2PerGameTeam           30.10
## pctFG2PerGameOpponent       28.45
## pctFGPerGameTeam            27.02
## pctFG3PerGameOpponent       20.67
## ptsPerGameOpponent          18.72
## drbPerGameTeam              18.34
## blkPerGameOpponent          17.56
## ageMeanMisc                 16.69
## astPerGameOpponent          16.12
## ptsPerGameTeam              13.41
```

```
plot(var.imp.knn, top = 20)
```



```r
knnval <- as.numeric(model.knn$bestTune)

##Conference Finals Analysis
##1st fold = validation set
MSEglm <- 0
Accuracyglm <- 0
Precisionglm <- 0
Recallglm <- 0
F1glm <- 0
AUCglm <- 0
ConfusMatglm <- vector(mode = "list", length = 4)
MSElda <- 0
Accuracylda <- 0
Precisionlda <- 0
Recalllda <- 0
F1lda <- 0
AUClda <- 0
ConfusMatlda <- vector(mode = "list", length = 4)
MSEknn <- 0
Accuracyknn <- 0
Precisionknn <- 0
Recallknn <- 0
F1knn <- 0
AUCknn <- 0
ConfusMatknn <- vector(mode = "list", length = 4)
ytrain <- ceiling((rbind(cbind(PerGameFold2[,3]),cbind(PerGameFold3[,3]),cbind(PerGameFold4[,3]))-2)/5)
```

```
xtrain <- rbind(PerGameFold2[,-3],PerGameFold3[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold1[,3]-2)/5)
xtest <- cbind(PerGameFold1[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~ Ranking + fg3mPerGameTeam + pctFG3PerGameOpponent +
                   pctFG3PerGameTeam + fgaPerGameTeam + fg3aPerGameTeam +
                   pctFG2PerGameOpponent + pctFG2PerGameTeam + ratioFTtoFGAOpponent,
                 data = datatrain, family = binomial)
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- 0
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- 1
  }
}
ConfusMatglm[[1]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[1]]
```

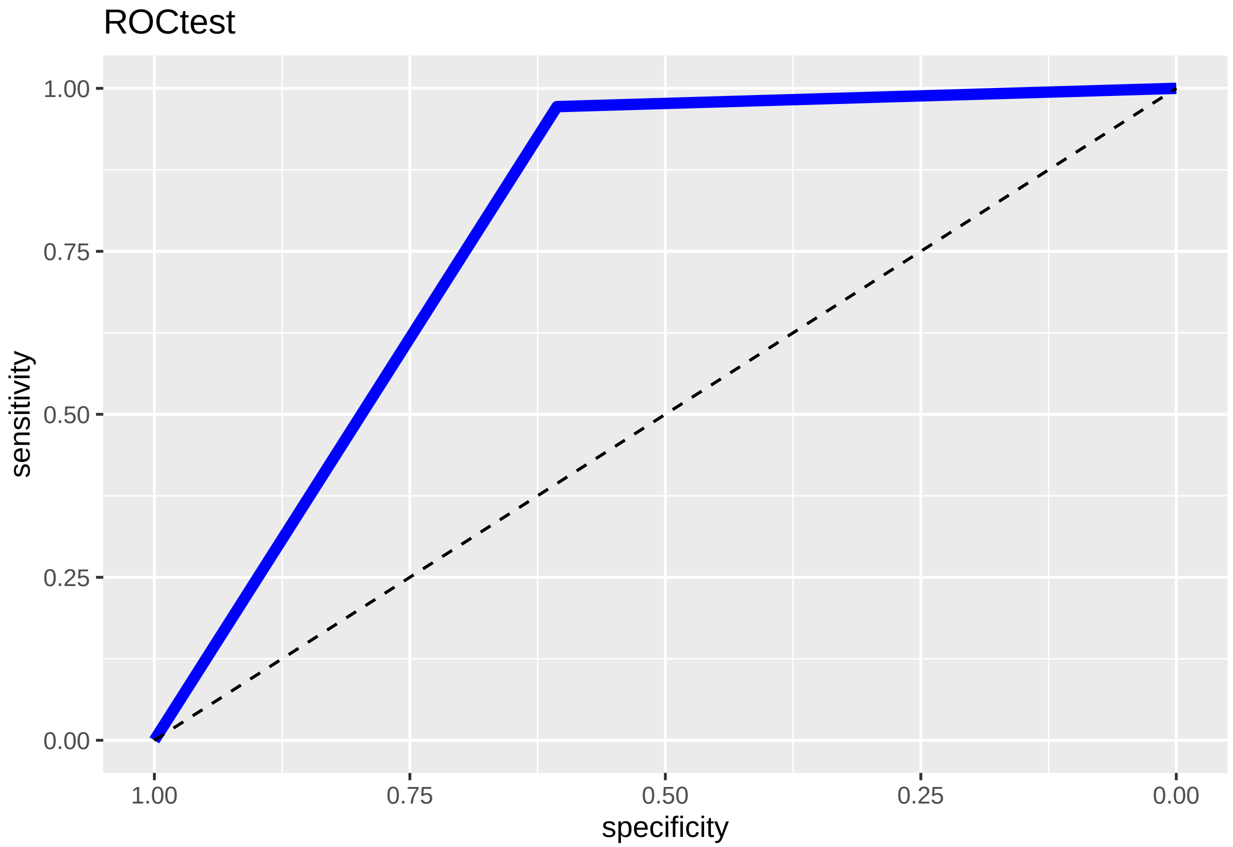
```
##       y_pred
## y_true   0   1
##      0  20  13
##      1   5 173
```

```
Accuracyglm <- Accuracyglm + ifelse(is.nan(Accuracy(factor(glmtest, levels=min(datatest$ytest):max(data
Precisionglm <- Precisionglm + ifelse(is.nan(Precision(factor(datatest$ytest, levels=min(datatest$ytest
Recallglm <- Recallglm + ifelse(is.nan(Recall(factor(datatest$ytest, levels=min(datatest$ytest):max(data
F1glm <- F1glm + ifelse(is.nan(F1_Score(factor(datatest$ytest, levels=min(datatest$ytest):max(datatest$
ROCtest <- roc(datatest$ytest, glmtest)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCglm <- AUCglm + AUC(glmtest, datatest$ytest)
MSEglm <- MSEglm + MSE(glmtest, datatest$ytest)

##Discriminant Models
##Linear Discriminant
model.lda <- lda(ytrain~Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
                   pctEFGTeamOppMisc, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.1559055 0.8440945
##
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.3226539  1.2491221    1.2500099         1.2447421        -1.0020840
## 1  0.1899747 -0.1901606   -0.1913447        -0.1906603         0.1196423
##
## Coefficients of linear discriminants:
##                         LD1
## Ranking           1.1610317
## winsTeam          0.7816898
## nrtgTeamMisc       -2.8658403
## marginVictoryTeam   2.1367528
## pctEFGTeamOppMisc   0.1213093
```

```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[1]] <-ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[1]]
```

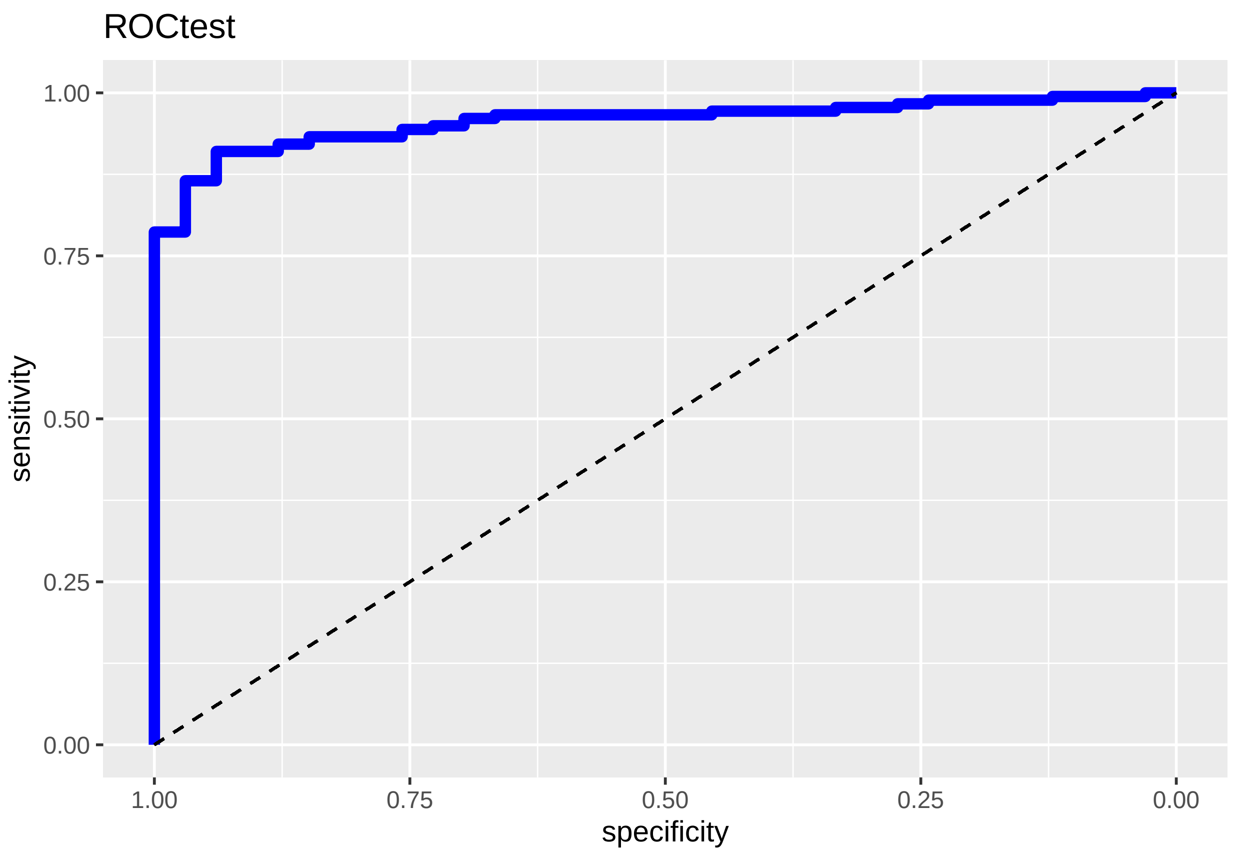
```
##       y_pred
## y_true   0   1
##      0  15  18
##      1   6 172
```

```
Accuracylda <- Accuracylda + ifelse(is.nan(Accuracy(ldatest$class, datatest$ytest)),0,Accuracy(ldatest$
Precisionlda <- Precisionlda + ifelse(is.nan(Precision(datatest$ytest, ldatest$class)),0,Precision(data
Recalllda <- Recalllda + ifelse(is.nan(Recall(datatest$ytest, ldatest$class)),0,Recall(datatest$ytest, 
F1lda <- F1lda + ifelse(is.nan(F1_Score(datatest$ytest, ldatest$class)),0,F1_Score(datatest$ytest, ldate
ROCtest <- roc(datatest$ytest, ldatest$posterior[,1])
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls > cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUClda <- AUClda + AUC(ldatest$class, datatest$ytest)
MSElda <- MSElda + MSE(as.numeric(ldatest$class), datatest$ytest)

##K Nearest Neighbours Model
model.knn <- knn3(formula = as.factor(ytrain)~ Ranking + winsTeam + nrtgTeamMisc +
                    marginVictoryTeam + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + drtgTeamMisc +
                    pctEFGTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameTeam +
                    pctFG2PerGameOpponent + pctFGPerGameTeam +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[1]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[1]]
```

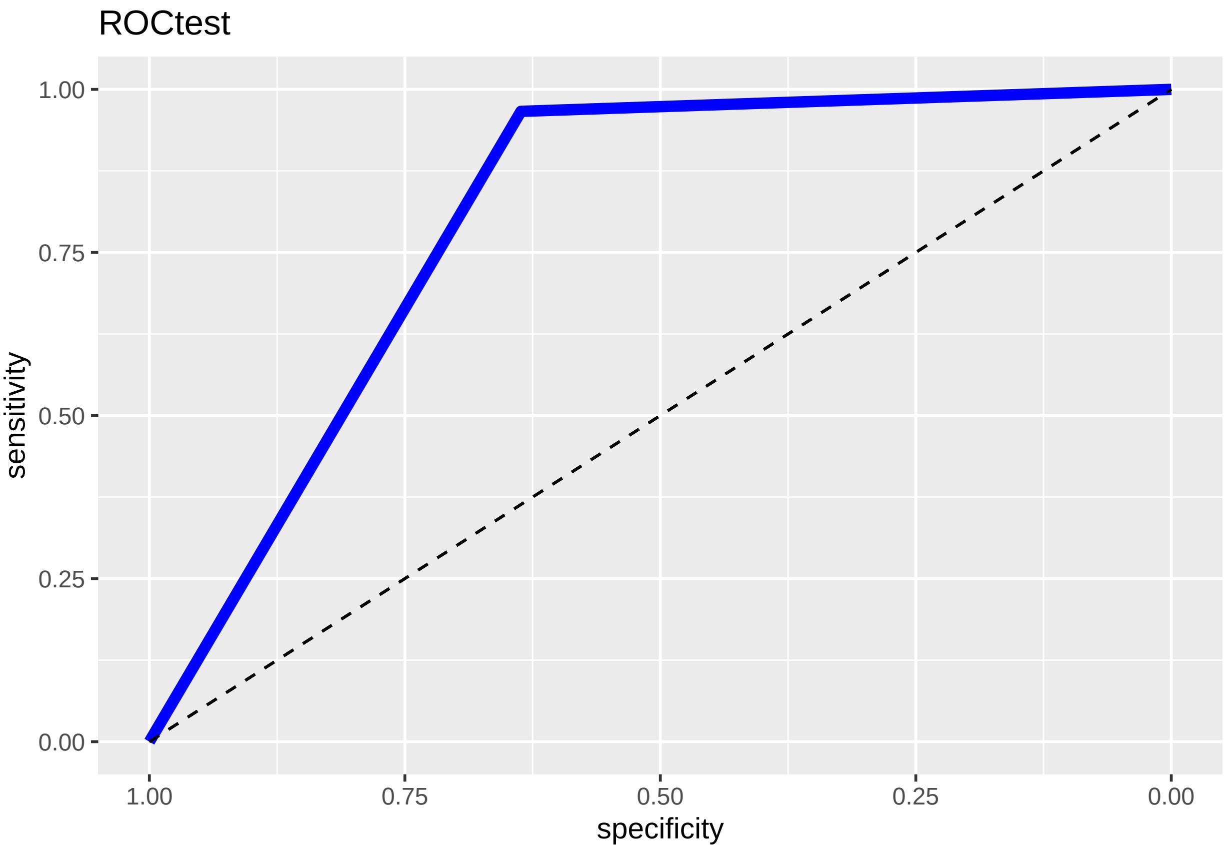
```
##         y_pred
## y_true   0   1
##      0  21  12
##      1   6 172
```

```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, knntes
F1knn <- F1knn + ifelse(is.nan(F1_Score(datatest$ytest, knntest)),0,F1_Score(datatest$ytest, knntest))
ROCtest <- roc(datatest$ytest, as.numeric(knntest)-1)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

##2nd fold = validation set
ytrain <- ceiling((rbind(cbind(PerGameFold1[,3]),cbind(PerGameFold3[,3]),cbind(PerGameFold4[,3]))-2)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold3[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold2[,3]-2)/5)
xtest <- cbind(PerGameFold2[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~ Ranking + fg3mPerGameTeam + pctFG3PerGameOpponent +
                   pctFG3PerGameTeam + fgaPerGameTeam + fg3aPerGameTeam +
                   pctFG2PerGameOpponent + pctFG2PerGameTeam + ratioFTtoFGAOpponent,
                 data = datatrain, family = binomial)
```

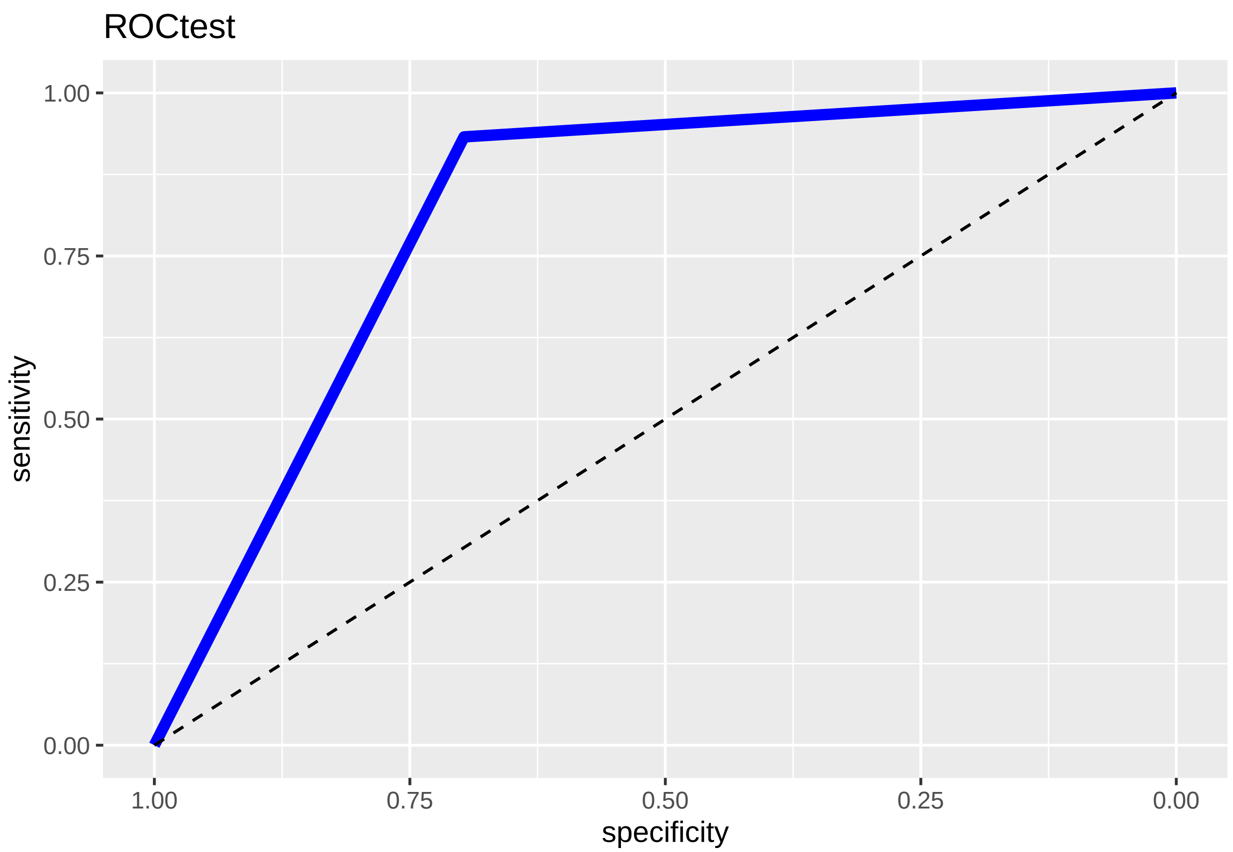
```r
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- 0
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- 1
  }
}
ConfusMatglm[[2]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[2]]
```

```
##       y_pred
## y_true   0   1
##      0  23  10
##      1  12 166
```

```r
Accuracyglm <- Accuracyglm + ifelse(is.nan(Accuracy(factor(glmtest, levels=min(datatest$ytest):max(data
Precisionglm <- Precisionglm + ifelse(is.nan(Precision(factor(datatest$ytest, levels=min(datatest$ytest)
Recallglm <- Recallglm + ifelse(is.nan(Recall(factor(datatest$ytest, levels=min(datatest$ytest):max(data
F1glm <- F1glm + ifelse(is.nan(F1_Score(factor(datatest$ytest, levels=min(datatest$ytest):max(datatest$
ROCtest <- roc(datatest$ytest, glmtest)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCglm <- AUCglm + AUC(glmtest, datatest$ytest)
MSEglm <- MSEglm + MSE(glmtest, datatest$ytest)

##Discriminant Models
##Linear Discriminant
model.lda <- lda(ytrain~Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
                   pctEFGTeamOppMisc, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.1559055 0.8440945
##
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.3397891  1.2536241    1.2461430         1.2426558        -0.9504802
## 1  0.2177494 -0.2094555   -0.2009227        -0.2007934         0.1297486
##
## Coefficients of linear discriminants:
##                          LD1
## Ranking           1.30484179
## winsTeam          0.60426063
## nrtgTeamMisc      -2.25200995
## marginVictoryTeam  1.77052299
## pctEFGTeamOppMisc  0.03883603
```

```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[2]] <- ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[2]]
```

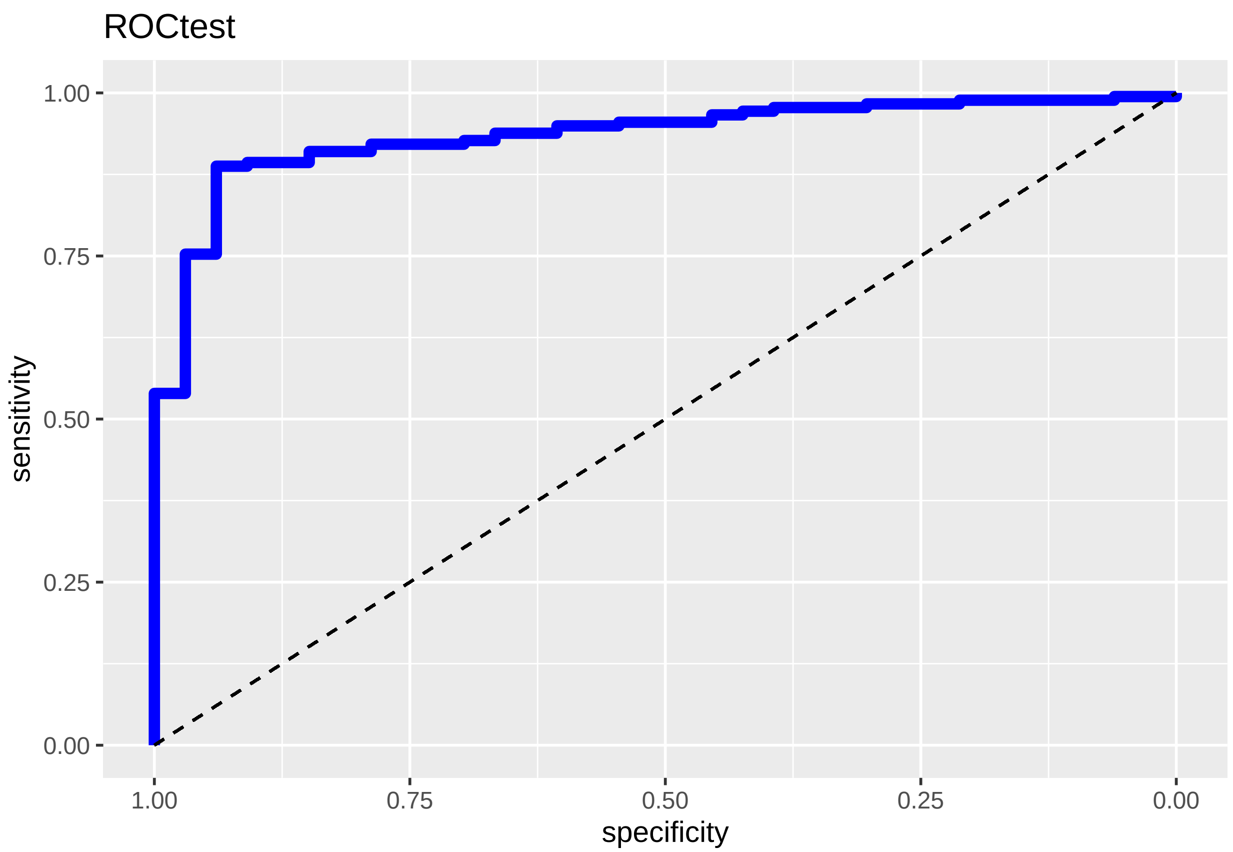
```
##       y_pred
## y_true   0   1
##      0  20  13
##      1   9 169
```

```
Accuracylda <- Accuracylda + ifelse(is.nan(Accuracy(ldatest$class, datatest$ytest)),0,Accuracy(ldatest$
Precisionlda <- Precisionlda + ifelse(is.nan(Precision(datatest$ytest, ldatest$class)),0,Precision(data
Recalllda <- Recalllda + ifelse(is.nan(Recall(datatest$ytest, ldatest$class)),0,Recall(datatest$ytest, l
F1lda <- F1lda + ifelse(is.nan(F1_Score(datatest$ytest, ldatest$class)),0,F1_Score(datatest$ytest, ldate
ROCtest <- roc(datatest$ytest, ldatest$posterior[,1])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUClda <- AUClda + AUC(ldatest$class, datatest$ytest)
MSElda <- MSElda + MSE(as.numeric(ldatest$class), datatest$ytest)

##K Nearest Neighbours Model
model.knn <- knn3(formula = as.factor(ytrain)~ Ranking + winsTeam + nrtgTeamMisc +
                    marginVictoryTeam + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + drtgTeamMisc +
                    pctEFGTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameTeam +
                    pctFG2PerGameOpponent + pctFGPerGameTeam +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[2]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[2]]
```

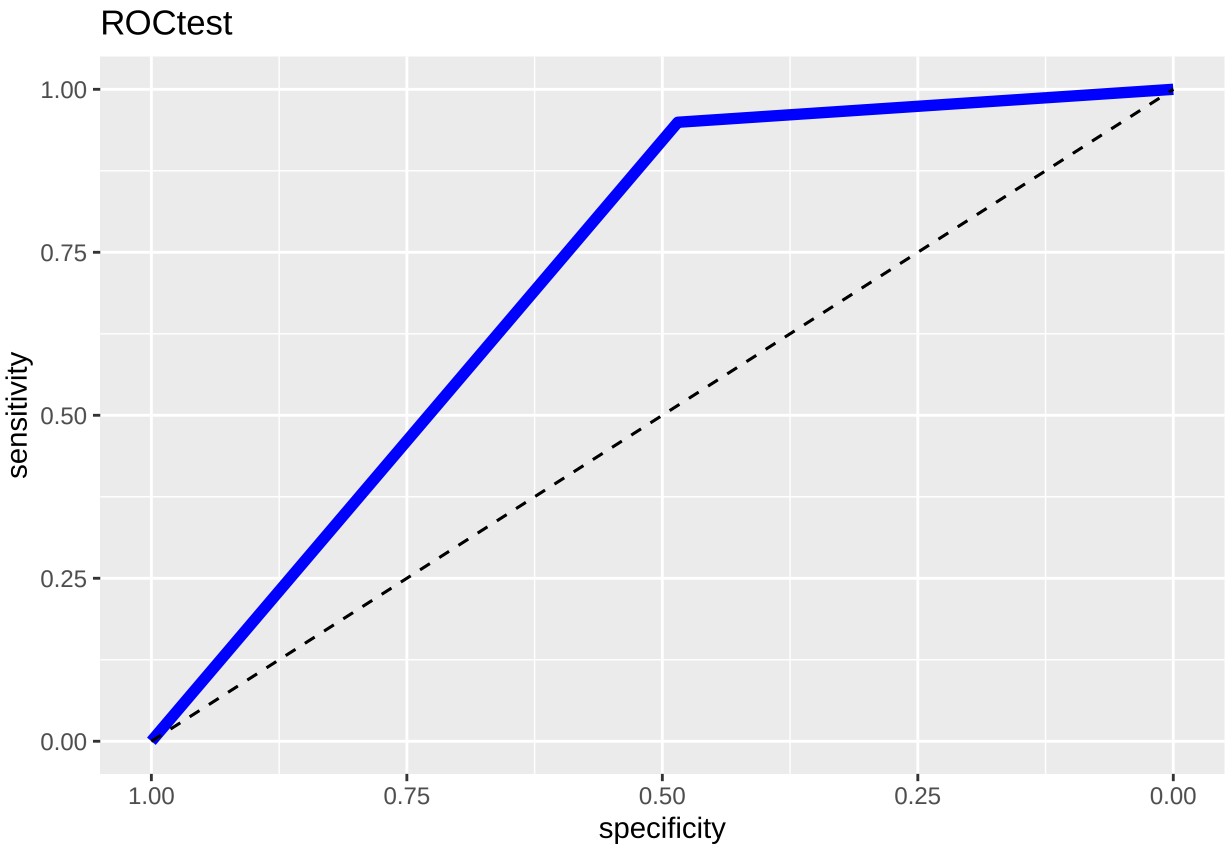
```
##       y_pred
## y_true   0   1
##      0  16  17
##      1   9 169
```

```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, knntes
F1knn <- F1knn + ifelse(is.nan(F1_Score(datatest$ytest, knntest)),0,F1_Score(datatest$ytest, knntest))
ROCtest <- roc(datatest$ytest, as.numeric(knntest)-1)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

##3rd fold = validation set
ytrain <- ceiling((rbind(cbind(PerGameFold1[,3]),cbind(PerGameFold2[,3]),cbind(PerGameFold4[,3]))-2)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold2[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold3[,3]-2)/5)
xtest <- cbind(PerGameFold3[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~ Ranking + fg3mPerGameTeam + pctFG3PerGameOpponent +
                   pctFG3PerGameTeam + fgaPerGameTeam + fg3aPerGameTeam +
                   pctFG2PerGameOpponent + pctFG2PerGameTeam + ratioFTtoFGAOpponent,
                 data = datatrain, family = binomial)
```

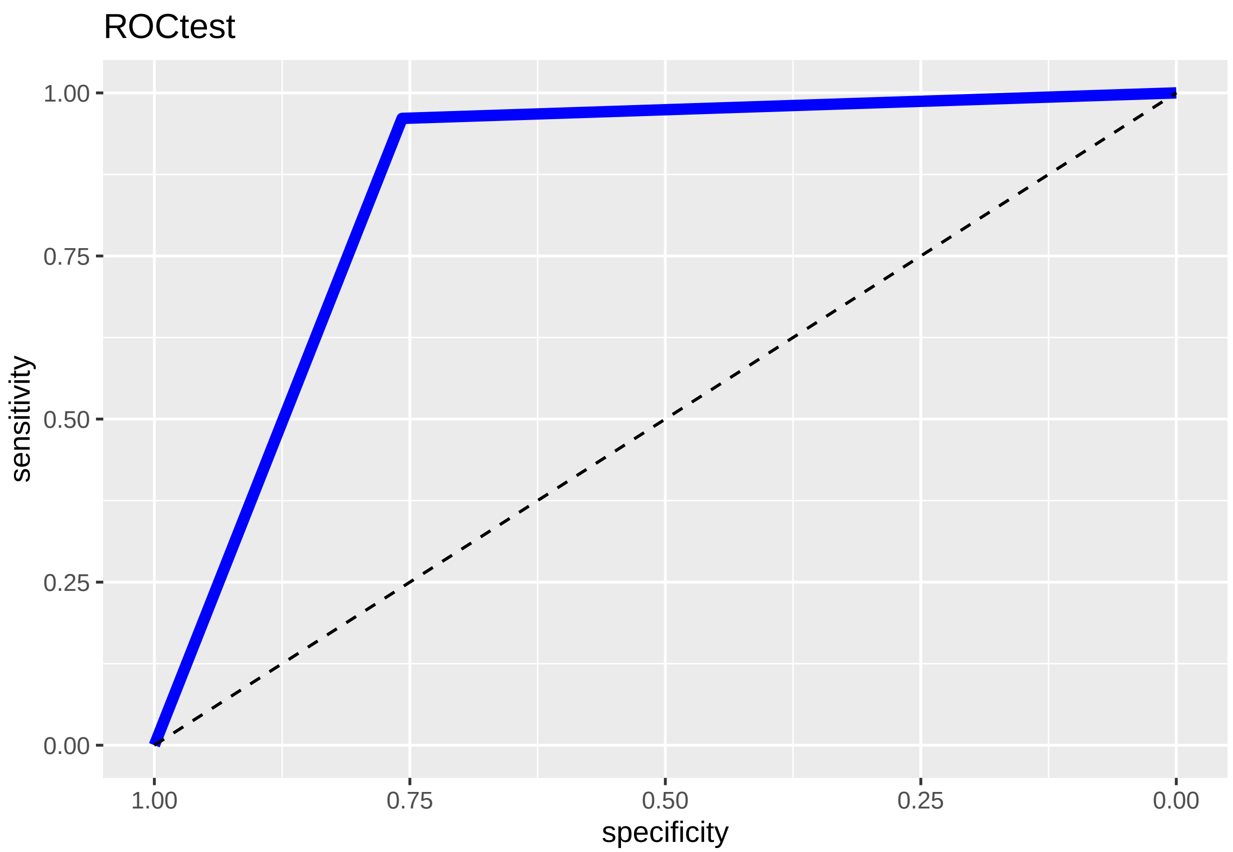
```
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- 0
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- 1
  }
}
ConfusMatglm[[3]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[3]]
```

```
##       y_pred
## y_true   0   1
##      0  25   8
##      1   7 172
```

```
Accuracyglm <- Accuracyglm + ifelse(is.nan(Accuracy(factor(glmtest, levels=min(datatest$ytest):max(datat
Precisionglm <- Precisionglm + ifelse(is.nan(Precision(factor(datatest$ytest, levels=min(datatest$ytest)
Recallglm <- Recallglm + ifelse(is.nan(Recall(factor(datatest$ytest, levels=min(datatest$ytest):max(data
F1glm <- F1glm + ifelse(is.nan(F1_Score(factor(datatest$ytest, levels=min(datatest$ytest):max(datatest$y
ROCtest <- roc(datatest$ytest, glmtest)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCglm <- AUCglm + AUC(glmtest, datatest$ytest)
MSEglm <- MSEglm + MSE(glmtest, datatest$ytest)

##Discriminant Models
##Linear Discriminant
model.lda <- lda(ytrain~Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
                   pctEFGTeamOppMisc, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.1561514 0.8438486
##
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.3235945  1.2604665    1.2523042         1.2476034        -0.9462515
## 1  0.1960977 -0.1870757   -0.1739053        -0.1724823         0.1149042
##
## Coefficients of linear discriminants:
##                           LD1
## Ranking            1.14476038
## winsTeam           0.27539092
## nrtgTeamMisc      -2.62815818
## marginVictoryTeam  2.36086648
## pctEFGTeamOppMisc  0.08222446
```

```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[3]] <- ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[3]]
```

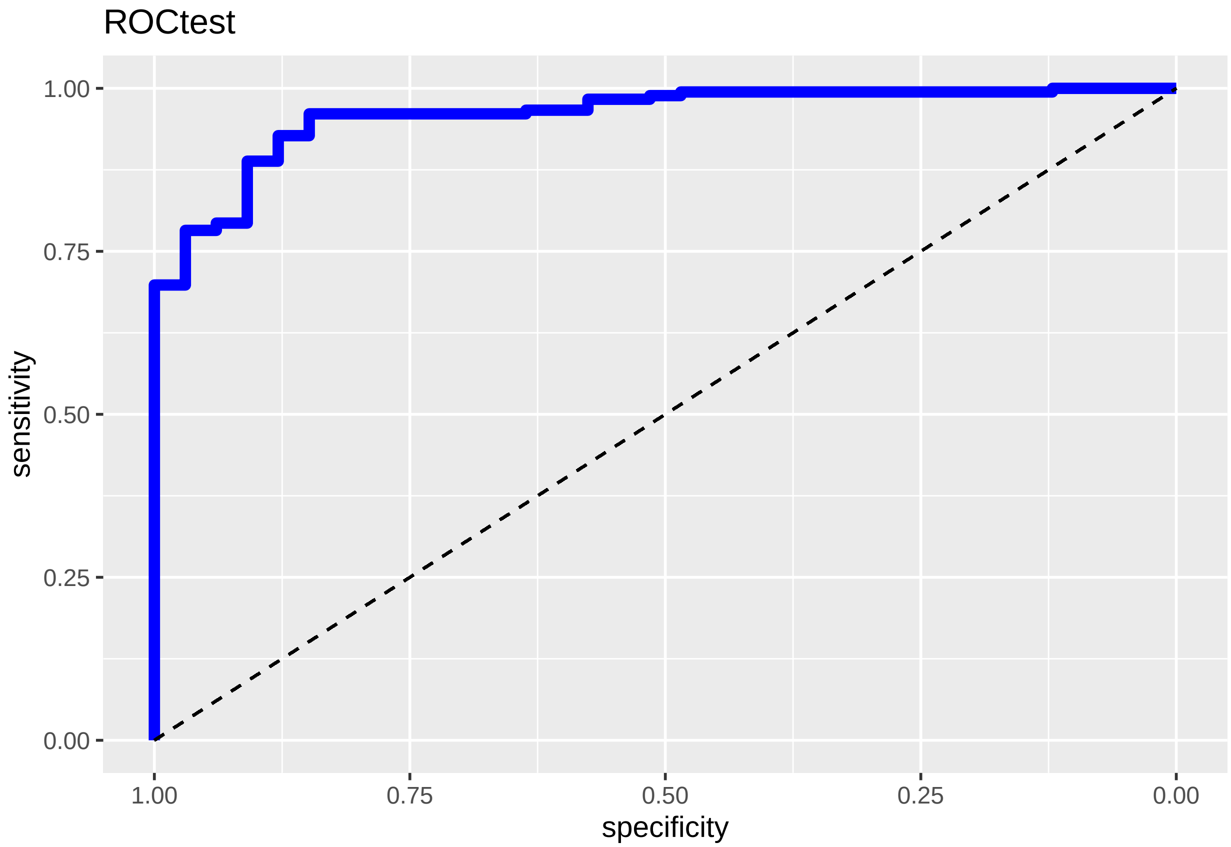
```
##       y_pred
## y_true   0   1
##      0  21  12
##      1   6 173
```

```
Accuracylda <- Accuracylda + ifelse(is.nan(Accuracy(ldatest$class, datatest$ytest)),0,Accuracy(ldatest$
Precisionlda <- Precisionlda + ifelse(is.nan(Precision(datatest$ytest, ldatest$class)),0,Precision(data
Recalllda <- Recalllda + ifelse(is.nan(Recall(datatest$ytest, ldatest$class)),0,Recall(datatest$ytest, l
F1lda <- F1lda + ifelse(is.nan(F1_Score(datatest$ytest, ldatest$class)),0,F1_Score(datatest$ytest, ldat
ROCtest <- roc(datatest$ytest, ldatest$posterior[,1])
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls > cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUClda <- AUClda + AUC(ldatest$class, datatest$ytest)
MSElda <- MSElda + MSE(as.numeric(ldatest$class), datatest$ytest)

##K Nearest Neighbours Model
model.knn <- knn3(formula = as.factor(ytrain)~ Ranking + winsTeam + nrtgTeamMisc +
                    marginVictoryTeam + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + drtgTeamMisc +
                    pctEFGTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameTeam +
                    pctFG2PerGameOpponent + pctFGPerGameTeam +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[3]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[3]]
```

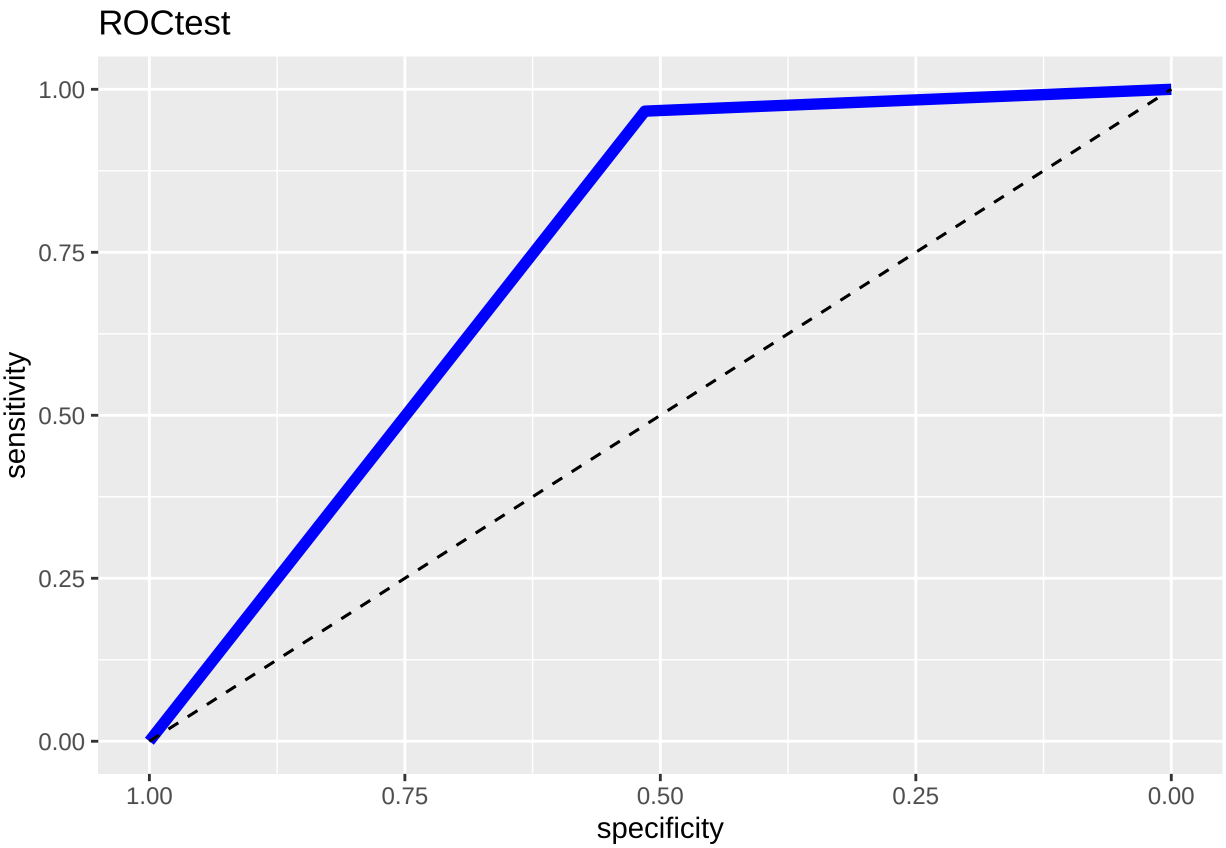
```
##       y_pred
## y_true   0   1
##      0  17  16
##      1   6 173
```

```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, knntes
F1knn <- F1knn + ifelse(is.nan(F1_Score(datatest$ytest, knntest)),0,F1_Score(datatest$ytest, knntest))
ROCtest <- roc(datatest$ytest, as.numeric(knntest)-1)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

##4th fold = validation set
ytrain <- ceiling((rbind(cbind(PerGameFold1[,3]),cbind(PerGameFold2[,3]),cbind(PerGameFold3[,3]))-2)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold2[,-3],PerGameFold3[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold4[,3]-2)/5)
xtest <- cbind(PerGameFold4[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~ Ranking + fg3mPerGameTeam + pctFG3PerGameOpponent +
                   pctFG3PerGameTeam + fgaPerGameTeam + fg3aPerGameTeam +
                   pctFG2PerGameOpponent + pctFG2PerGameTeam + ratioFTtoFGAOpponent,
                 data = datatrain, family = binomial)
```

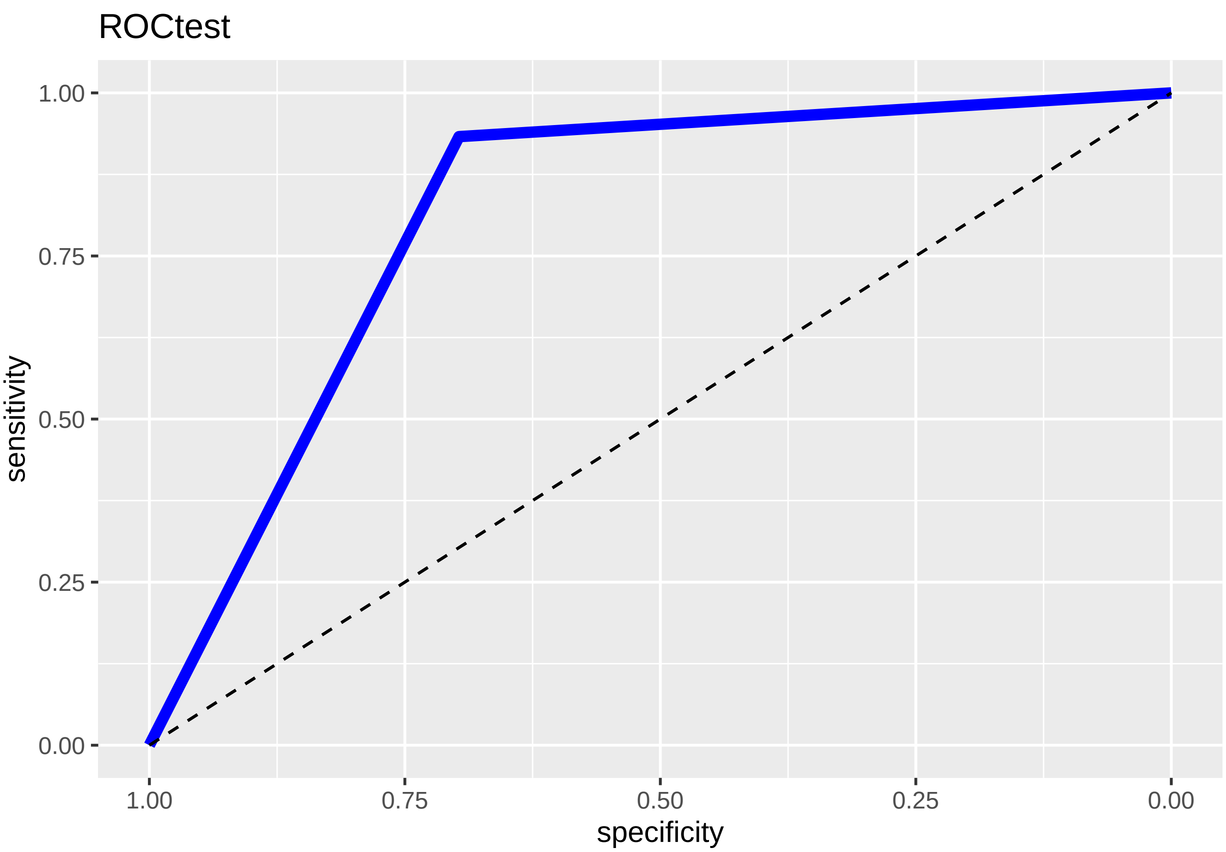
```
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- 0
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- 1
  }
}
ConfusMatglm[[4]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[4]]
```

```
##       y_pred
## y_true   0   1
##      0  23  10
##      1  12 167
```

```
Accuracyglm <- Accuracyglm + ifelse(is.nan(Accuracy(factor(glmtest, levels=min(datatest$ytest):max(data
Precisionglm <- Precisionglm + ifelse(is.nan(Precision(factor(datatest$ytest, levels=min(datatest$ytest)
Recallglm <- Recallglm + ifelse(is.nan(Recall(factor(datatest$ytest, levels=min(datatest$ytest):max(data
F1glm <- F1glm + ifelse(is.nan(F1_Score(factor(datatest$ytest, levels=min(datatest$ytest):max(datatest$y
ROCtest <- roc(datatest$ytest, glmtest)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCglm <- AUCglm + AUC(glmtest, datatest$ytest)
MSEglm <- MSEglm + MSE(glmtest, datatest$ytest)

##Discriminant Models
##Linear Discriminant
model.lda <- lda(ytrain~Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
                   pctEFGTeamOppMisc, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.1561514 0.8438486
##
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.3425451  1.2871808     1.288302         1.2827166        -0.9789355
## 1  0.2137373 -0.2104686    -0.195169        -0.1944125         0.1096967
##
## Coefficients of linear discriminants:
##                          LD1
## Ranking           1.18166176
## winsTeam          0.34023314
## nrtgTeamMisc      -2.01851208
## marginVictoryTeam  1.66172984
## pctEFGTeamOppMisc  0.01728334
```

```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[4]] <- ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[4]]
```

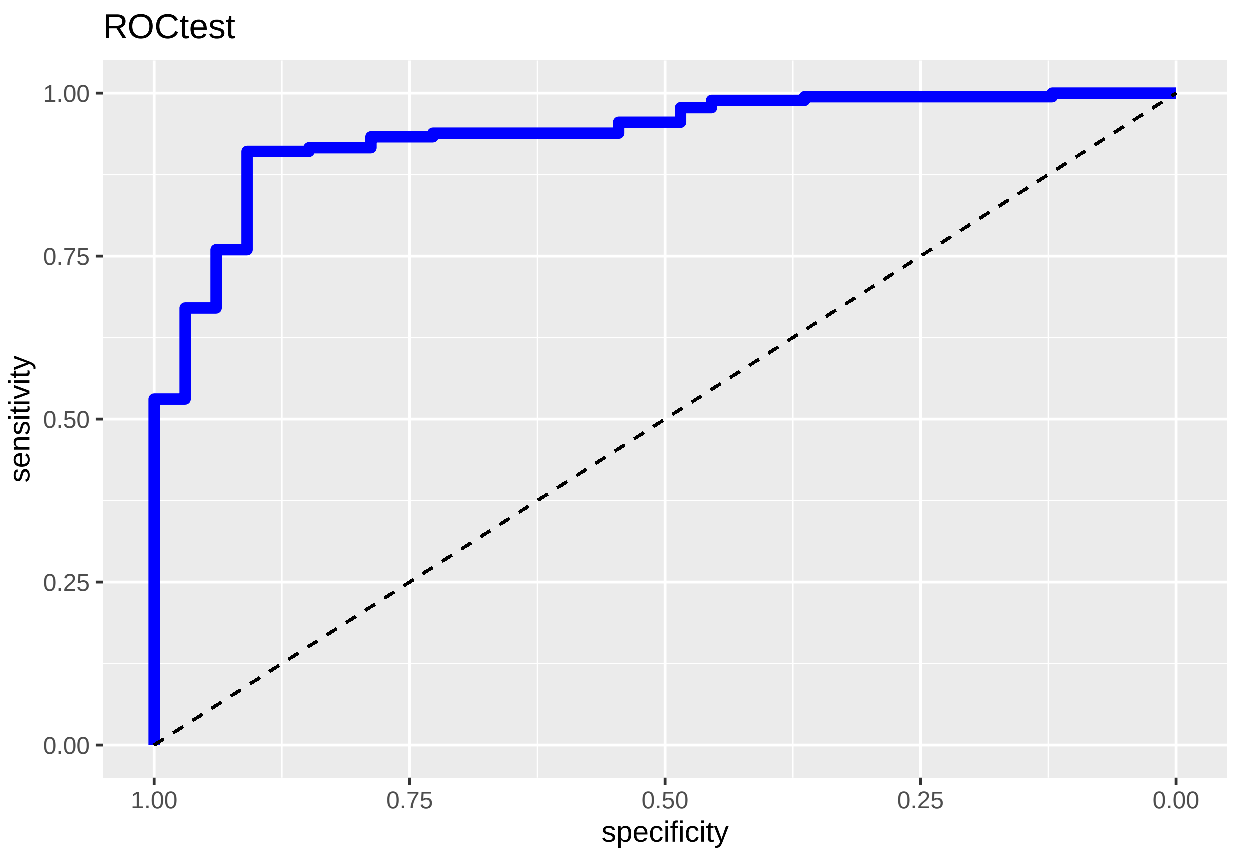
```
##       y_pred
## y_true   0   1
##      0  18  15
##      1   8 171
```

```
Accuracylda <- Accuracylda + ifelse(is.nan(Accuracy(ldatest$class, datatest$ytest)),0,Accuracy(ldatest$c
Precisionlda <- Precisionlda + ifelse(is.nan(Precision(datatest$ytest, ldatest$class)),0,Precision(datat
Recalllda <- Recalllda + ifelse(is.nan(Recall(datatest$ytest, ldatest$class)),0,Recall(datatest$ytest, l
F1lda <- F1lda + ifelse(is.nan(F1_Score(datatest$ytest, ldatest$class)),0,F1_Score(datatest$ytest, ldate
ROCtest <- roc(datatest$ytest, ldatest$posterior[,1])
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls > cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```r
AUClda <- AUClda + AUC(ldatest$class, datatest$ytest)
MSElda <- MSElda + MSE(as.numeric(ldatest$class), datatest$ytest)

##K Nearest Neighbours Model
model.knn <- knn3(formula = as.factor(ytrain)~ Ranking + winsTeam + nrtgTeamMisc +
                    marginVictoryTeam + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + drtgTeamMisc +
                    pctEFGTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameTeam +
                    pctFG2PerGameOpponent + pctFGPerGameTeam +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[4]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[4]]
```

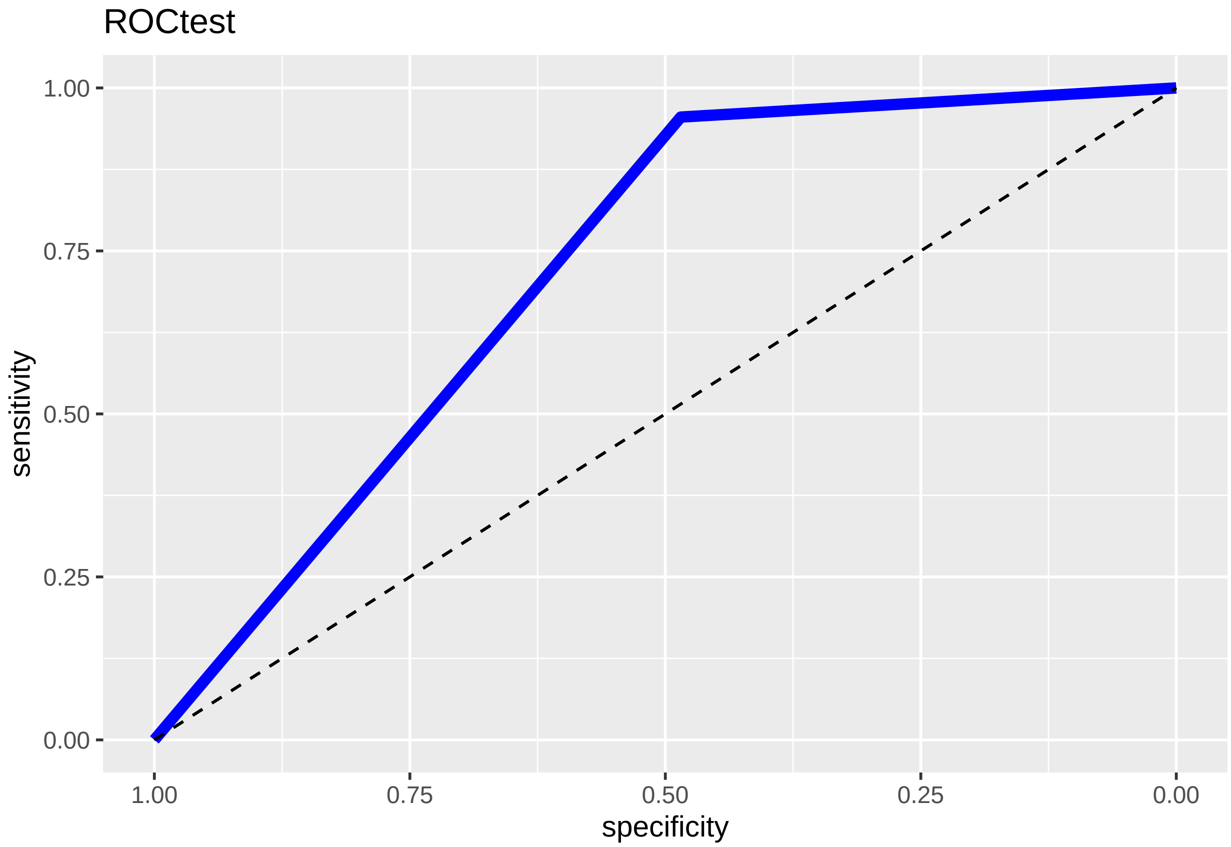
```
##       y_pred
## y_true   0   1
##      0  16  17
##      1   8 171
```

```r
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, knntes
F1knn <- F1knn + ifelse(is.nan(F1_Score(datatest$ytest, knntest)),0,F1_Score(datatest$ytest, knntest))
ROCtest <- roc(datatest$ytest, as.numeric(knntest)-1)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
ggroc(ROCtest, colour = "blue", linetype = 1, size =2) +
  ggtitle("ROCtest") +
  geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1), colour = "black", linetype = 2) +
  theme_gray()
```



```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

##Let us take a look at our metrics for each model
##Logistic Regression
MSEglm/4
```

```
## [1] 0.09102544
```

```
Accuracyglm/4
```

```
## [1] 0.9089746
```

```
Precisionglm/4
```

```
## [1] 0.7238839
```

```
Recallglm/4
```

```
## [1] 0.6893939
```

```
F1glm/4
```

```
## [1] 0.7029568
```

```
AUCglm/4
```

```
## [1] 0.8194906
```

```
ConfusMatglm
```

```
## [[1]]
##        y_pred
## y_true   0   1
##      0  20  13
##      1   5 173
##
## [[2]]
##        y_pred
## y_true   0   1
##      0  23  10
##      1  12 166
##
## [[3]]
##        y_pred
## y_true   0   1
##      0  25   8
##      1   7 172
##
## [[4]]
##        y_pred
## y_true   0   1
##      0  23  10
##      1  12 167
```

```
##Linear Discriminant
MSElda/4
```

```
## [1] 1.171426
```

```
Accuracylda/4
```

```
## [1] 0.8971486
```

```
Precisionlda/4
```

```
## [1] 0.7185066
```

```
Recalllda/4
```

```
## [1] 0.5606061
```

```
F1lda/4
```

```
## [1] 0.6277216
```

```
AUClda/4
```

```
## [1] 0.7599928
```

```
ConfusMatlda
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0  15  18
##      1   6 172
## 
## [[2]]
##       y_pred
## y_true   0   1
##      0  20  13
##      1   9 169
## 
## [[3]]
##       y_pred
## y_true   0   1
##      0  21  12
##      1   6 173
## 
## [[4]]
##       y_pred
## y_true   0   1
##      0  18  15
##      1   8 171
```

```
##K Nearest Neighbours
MSEknn/4
```

```
## [1] 0.1075572
```

```
Accuracyknn/4
```

```
## [1] 0.8924428
```

```
Precisionknn/4
```

```
## [1] 0.7058937
```

```
Recallknn/4
```

```
## [1] 0.530303
```

```
F1knn/4
```

```
## [1] 0.6050676
```

```
AUCknn/4
```

```
## [1] 0.7448413
```

```
ConfusMatknn
```

```
## [[1]]
##        y_pred
## y_true   0   1
##      0  21  12
##      1   6 172
## 
## [[2]]
##        y_pred
## y_true   0   1
##      0  16  17
##      1   9 169
## 
## [[3]]
##        y_pred
## y_true   0   1
##      0  17  16
##      1   6 173
## 
## [[4]]
##        y_pred
## y_true   0   1
##      0  16  17
##      1   8 171
```

```
##2020 Season Predictions
ytrain <- ceiling((MasterPerGame$finish-2)/5)
xtrain <- MasterPerGame[,-3]
datatrain <- cbind(ytrain, xtrain)
xtest <- MasterPerGame2020

##Logistic Regression
model.glm <- glm(ytrain~ Ranking + fg3mPerGameTeam + pctFG3PerGameOpponent +
                   pctFG3PerGameTeam + fgaPerGameTeam + fg3aPerGameTeam +
                   pctFG2PerGameOpponent + pctFG2PerGameTeam + ratioFTtoFGAOpponent,
                 data = datatrain, family = binomial)
glmtest <- predict(model.glm, xtest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- 0
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- 1
  }
}
glmtest
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  1  1  1  1  1  1  1  1  1  1  1  1  0  0  1  1  0  1  1  1  1  1  1  1  1  1
## 27 28 29 30
##  1  0  1  1
```

```
for (i in 1:length(glmtest)) {
  if(glmtest[i] == 0){
    print(as.character(NBASalaryAnalysisData2020$Team[i]))
  }
}
```

```
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Milwaukee Bucks"
## [1] "Toronto Raptors"
```

```
##Logistic Regression Model suggests that the LA Clippers, Los Angeles Lakers,
##Milwaukee Bucks and Toronto Raptors are Conference Finals level teams

##Discriminant Analysis
##Linear Discriminant
model.lda <- lda(ytrain~Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
                   pctEFGTeamOppMisc, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc, data = datatrain)
##
## Prior probabilities of groups:
```

```
##         0         1 
## 0.1371158 0.8628842 
## 
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.3643393  1.3027817    1.2786842         1.2761289        -1.0030238
## 1  0.2167991 -0.2070174   -0.2031882        -0.2027821         0.1593846
## 
## Coefficients of linear discriminants:
##                         LD1
## Ranking            1.0959272
## winsTeam           0.2230959
## nrtgTeamMisc      -2.3074136
## marginVictoryTeam  2.0627518
## pctEFGTeamOppMisc  0.1158523
```

```r
ldatest <- predict(model.lda, xtest)
ldatest$class
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1
## Levels: 0 1
```

```r
for (i in 1:length(ldatest$class)) {
  if(ldatest$class[i] == 0){
    print(as.character(NBASalaryAnalysisData2020$Team[i]))
  }
}
```

```
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Milwaukee Bucks"
## [1] "Toronto Raptors"
```

```r
##Discriminant Analysis Model suggests the LA Clippers, Los Angeles Lakers,
##Milwaukee Bucks and Toronto Raptors are Conference Finals level teams

##K Nearest Neighbours
model.knn <- knn3(formula = as.factor(ytrain)~ Ranking + winsTeam + nrtgTeamMisc +
                    marginVictoryTeam + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + drtgTeamMisc +
                    pctEFGTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameTeam +
                    pctFG2PerGameOpponent + pctFGPerGameTeam +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, xtest, type = "class")
knntest
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## Levels: 0 1
```

```
for (i in 1:length(knntest)) {
  if(knntest[i] == 0){
    print(as.character(NBASalaryAnalysisData2020$Team[i]))
  }
}
```

```
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
```

```
##K Nearest Neighbours suggests that the LA Clippers and Los Angeles Lakers
##are Conference Finals level teams
```