# NBAAnalysisSecondRoundClassificationModel.R

dpesl

2020-05-17

```
##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
```

```
##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
##Second Round Feature Selection
ytrain <- ceiling((MasterPerGame$finish-3)/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
```

```
model.glm
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (4 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD  KappaSD Selected
##          5   0.8794 0.6942   0.023822 0.058100
##          6   0.8782 0.6914   0.021809 0.053692
##          7   0.8782 0.6914   0.021809 0.053692
##          8   0.8853 0.7084   0.022536 0.056705
##          9   0.8853 0.7091   0.022536 0.057423
##         10   0.8853 0.7098   0.022536 0.058147
##         11   0.8841 0.7067   0.019726 0.049336
##         12   0.8841 0.7083   0.021564 0.053042
##         13   0.8865 0.7133   0.021683 0.054729        *
##         14   0.8829 0.7050   0.024485 0.060640
##         15   0.8829 0.7050   0.024485 0.060640
##         16   0.8829 0.7047   0.020058 0.051124
##         17   0.8806 0.6982   0.013326 0.031364
##         18   0.8782 0.6913   0.012763 0.030781
##         19   0.8782 0.6904   0.015940 0.039333
##         20   0.8770 0.6891   0.018818 0.042806
##         21   0.8711 0.6736   0.010808 0.020196
##         22   0.8723 0.6761   0.010502 0.020352
##         23   0.8735 0.6796   0.011476 0.022588
##         24   0.8700 0.6699   0.008439 0.016915
##         25   0.8735 0.6786   0.008372 0.015757
##         62   0.8629 0.6517   0.004134 0.009414
##
## The top 5 variables (out of 13):
##    Ranking, fg3mPerGameOpponent, stlPerGameTeam, fg3mPerGameTeam, pctTOVOpponentMisc
```

```
model.glm$optVariables
```

```
##  [1] "Ranking"             "fg3mPerGameOpponent" "stlPerGameTeam"
##  [4] "fg3mPerGameTeam"     "pctTOVOpponentMisc"  "ortgTeamMisc"
##  [7] "winsTeam"            "fg2mPerGameTeam"     "stlPerGameOpponent"
## [10] "fg3aPerGameOpponent" "pctFTPerGameTeam"    "fg2aPerGameTeam"
## [13] "pctFGPerGameTeam"
```

```
##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   0.8688 0.6820    0.01939 0.04627
##          6   0.8794 0.7077    0.02602 0.06042
##          7   0.8782 0.7022    0.02712 0.06284
##          8   0.8853 0.7201    0.03067 0.07017
##          9   0.8888 0.7274    0.02762 0.06415
##         10   0.8830 0.7146    0.02178 0.04995
##         11   0.8841 0.7171    0.02115 0.04816
##         12   0.8865 0.7225    0.02932 0.06782
##         13   0.8900 0.7322    0.02992 0.06932        *
##         14   0.8865 0.7215    0.03273 0.07758
##         15   0.8818 0.7109    0.02373 0.05686
##         16   0.8818 0.7097    0.02177 0.04981
##         17   0.8818 0.7095    0.02031 0.04781
##         18   0.8841 0.7154    0.02479 0.05908
##         19   0.8806 0.7059    0.02310 0.05789
##         20   0.8806 0.7063    0.02078 0.04858
##         21   0.8782 0.7012    0.02178 0.05159
##         22   0.8853 0.7197    0.02990 0.07331
##         23   0.8841 0.7173    0.03116 0.07553
##         24   0.8794 0.7061    0.02760 0.06585
##         25   0.8818 0.7117    0.02954 0.07121
##         62   0.8676 0.6746    0.01030 0.02344
##
## The top 5 variables (out of 13):
##    Ranking, winsTeam, nrtgTeamMisc, marginVictoryTeam, pctEFGTeamOppMisc
```

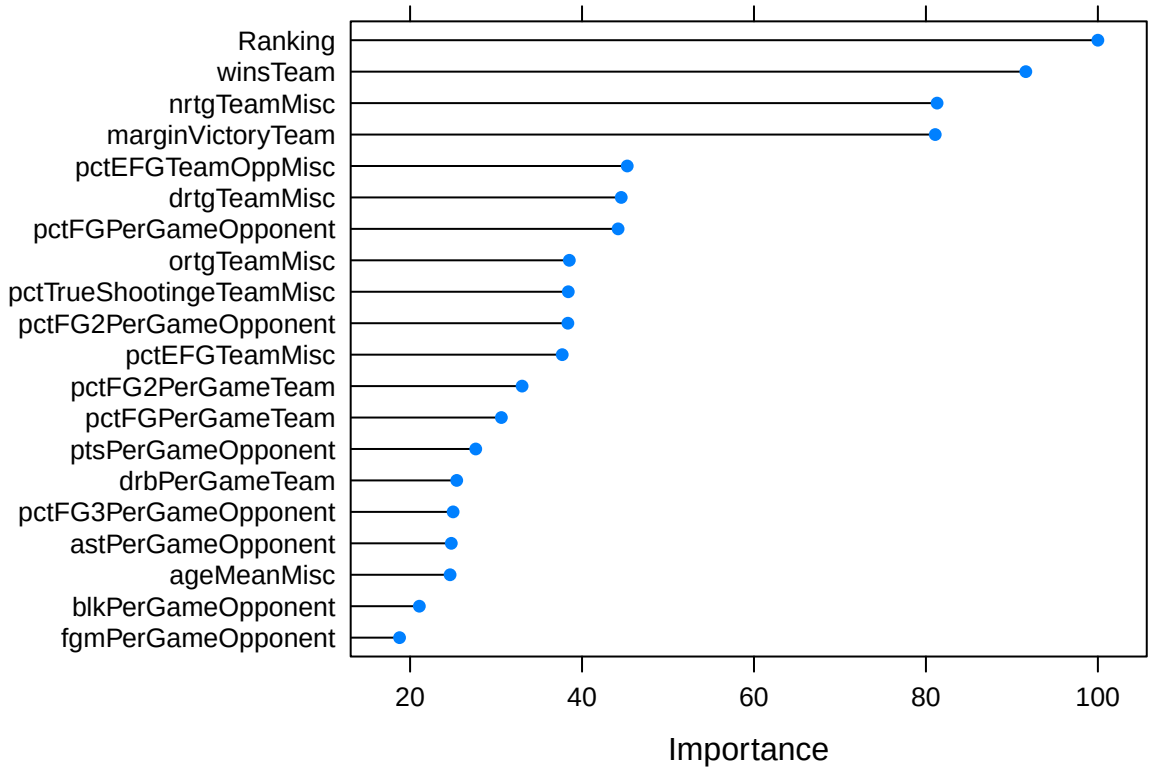
```
model.lda$optVariables
```

```
##  [1] "Ranking"                  "winsTeam"
##  [3] "nrtgTeamMisc"             "marginVictoryTeam"
##  [5] "pctEFGTeamOppMisc"        "drtgTeamMisc"
##  [7] "pctFGPerGameOpponent"     "ortgTeamMisc"
##  [9] "pctFG2PerGameOpponent"    "pctTrueShootingeTeamMisc"
## [11] "pctEFGTeamMisc"           "pctFG2PerGameTeam"
## [13] "pctFGPerGameTeam"
```

```
##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                    91.62
## nrtgTeamMisc                81.30
## marginVictoryTeam           81.08
## pctEFGTeamOppMisc           45.26
## drtgTeamMisc                44.56
## pctFGPerGameOpponent        44.19
## ortgTeamMisc                38.52
## pctTrueShootingeTeamMisc    38.40
## pctFG2PerGameOpponent       38.35
## pctEFGTeamMisc              37.69
## pctFG2PerGameTeam           33.03
## pctFGPerGameTeam            30.62
## ptsPerGameOpponent          27.64
## drbPerGameTeam              25.42
## pctFG3PerGameOpponent       25.00
## astPerGameOpponent          24.79
## ageMeanMisc                 24.65
## blkPerGameOpponent          21.07
## fgmPerGameOpponent          18.77
```

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




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

##Second Round 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]))-3)/5)
```

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

##Logistic Regression
model.glm <- glm(ytrain~Ranking + stlPerGameTeam + fg3mPerGameOpponent +
                   fg3mPerGameTeam + ortgTeamMisc + pctTOVOpponentMisc +
                   fg3aPerGameOpponent + winsTeam + stlPerGameOpponent +
                   fg2mPerGameTeam + pctFTPerGameTeam + fg2aPerGameTeam +
                   pctFGPerGameTeam, 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]]
```

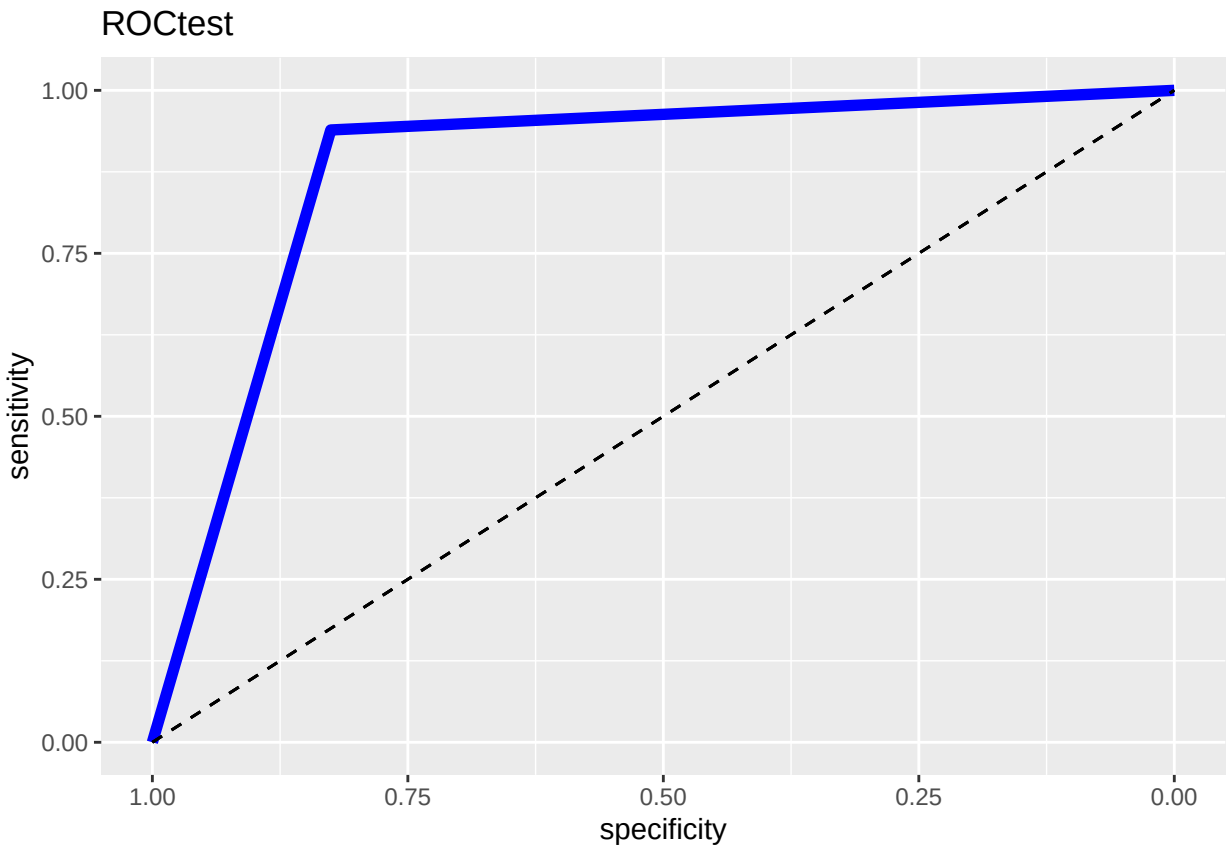
```
##        y_pred
## y_true   0   1
##      0  52  11
##      1   9 139
```

```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$ytest):max(datatest$y
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 + drtgTeamMisc + pctFGPerGameOpponent +
                   ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam,
                 data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc + drtgTeamMisc + pctFGPerGameOpponent +
##     ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.2976378 0.7023622
##
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.1255599  1.0544102     1.012422         1.0092686        -0.8359338
## 1  0.4116918 -0.3980864    -0.381519        -0.3805297         0.2755907
```

```
##   drtgTeamMisc pctFGPerGameOpponent ortgTeamMisc pctFG2PerGameOpponent
## 0   -0.8631366            -0.817451    0.7143863            -0.7910250
## 1    0.3108187             0.273092   -0.2823564             0.2420244
##   pctTrueShootingeTeamMisc pctEFGTeamMisc pctFG2PerGameTeam pctFGPerGameTeam
## 0                0.6510718      0.6794782         0.6629366        0.6192552
## 1               -0.2668333     -0.2650884        -0.2506140       -0.2395832
##
## Coefficients of linear discriminants:
##                                 LD1
## Ranking                  1.44782979
## winsTeam                 0.05986326
## nrtgTeamMisc            -1.12808019
## marginVictoryTeam        1.25643909
## pctEFGTeamOppMisc        0.76942240
## drtgTeamMisc            -0.02009813
## pctFGPerGameOpponent    -0.39682623
## ortgTeamMisc             0.04897121
## pctFG2PerGameOpponent   -0.24499050
## pctTrueShootingeTeamMisc -0.03301022
## pctEFGTeamMisc          -0.08350136
## pctFG2PerGameTeam       -0.15034834
## pctFGPerGameTeam         0.16654853
```

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

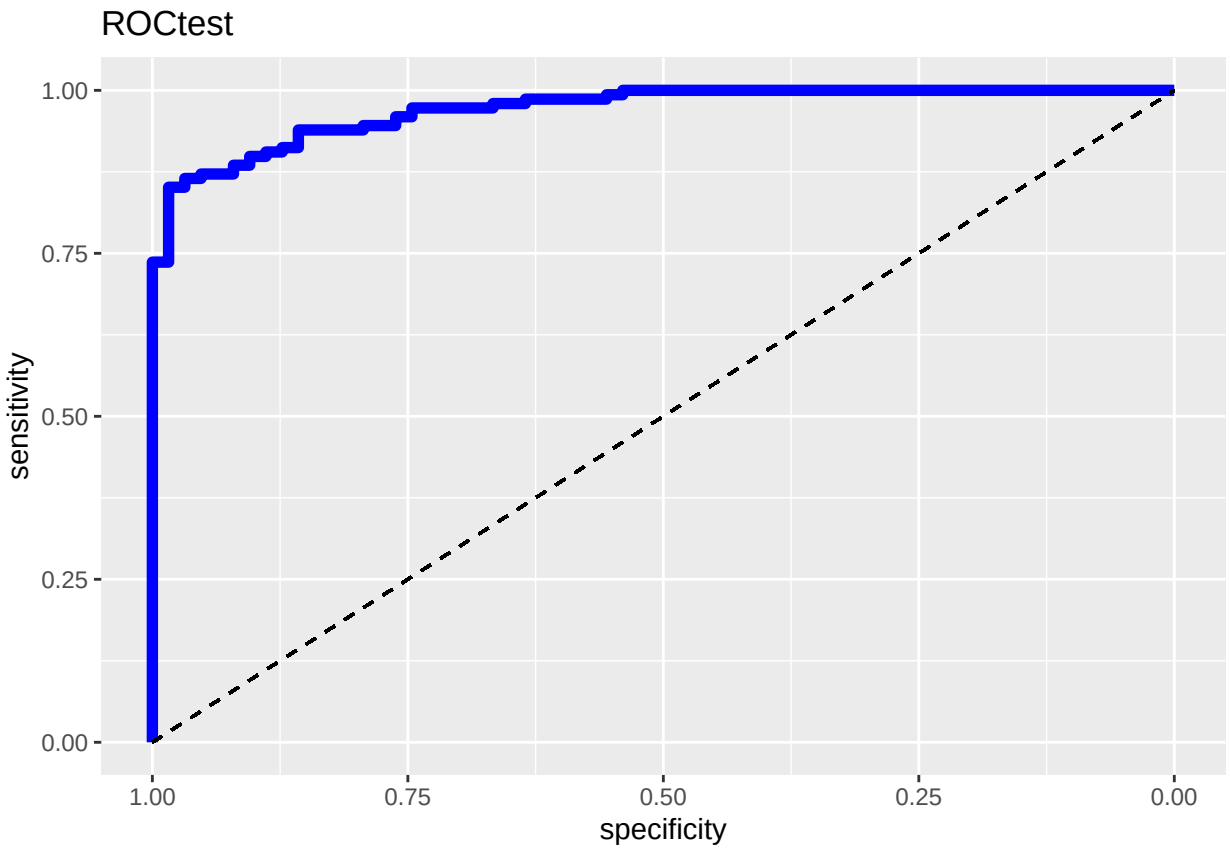
```
##       y_pred
## y_true   0   1
##      0  54   9
##      1  13 135
```

```
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 +
                    pctEFGTeamOppMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + pctFG2PerGameOpponent +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + pctFGPerGameTeam +
                    ptsPerGameOpponent + drbPerGameTeam +
                    pctFG3PerGameOpponent + astPerGameOpponent +
                    ageMeanMisc + blkPerGameOpponent,
                  data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[1]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[1]]
```

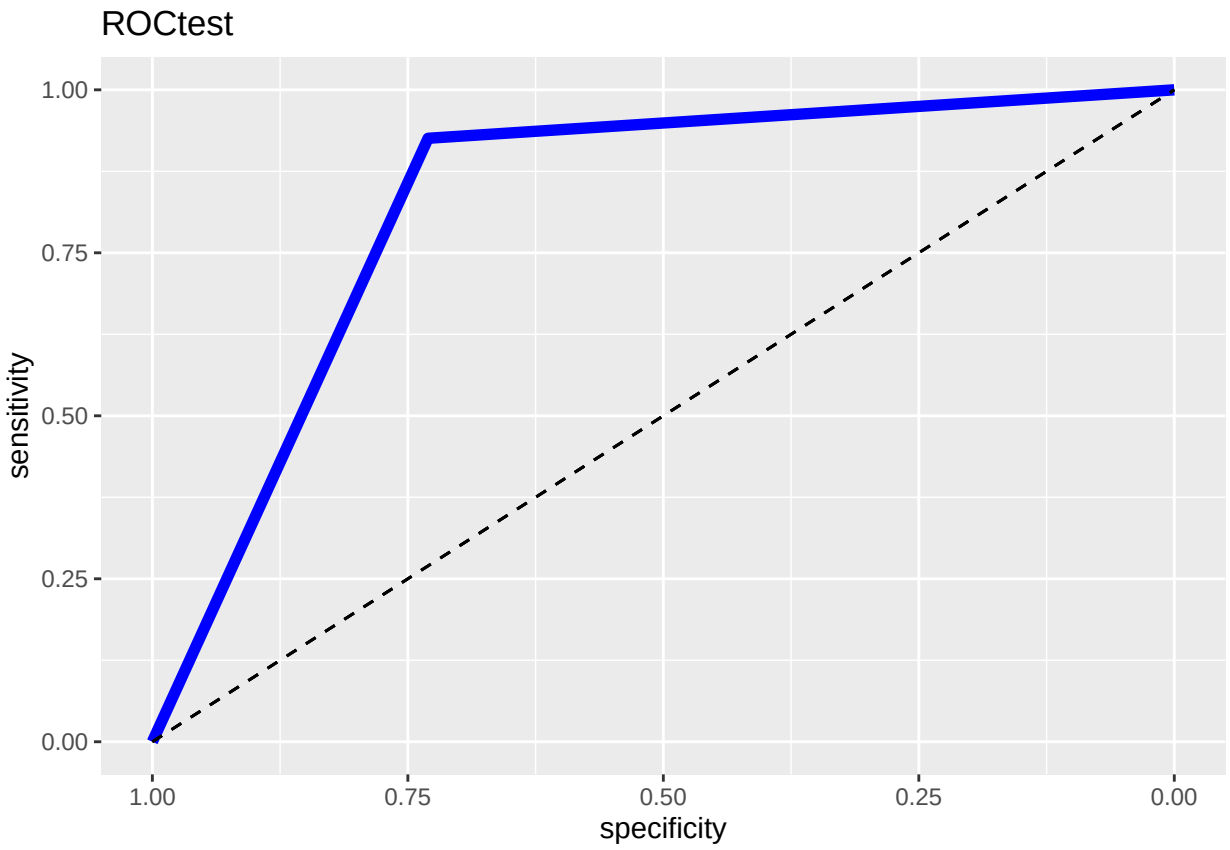
```
##        y_pred
## y_true   0   1
##      0  46  17
##      1  11 137
```

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

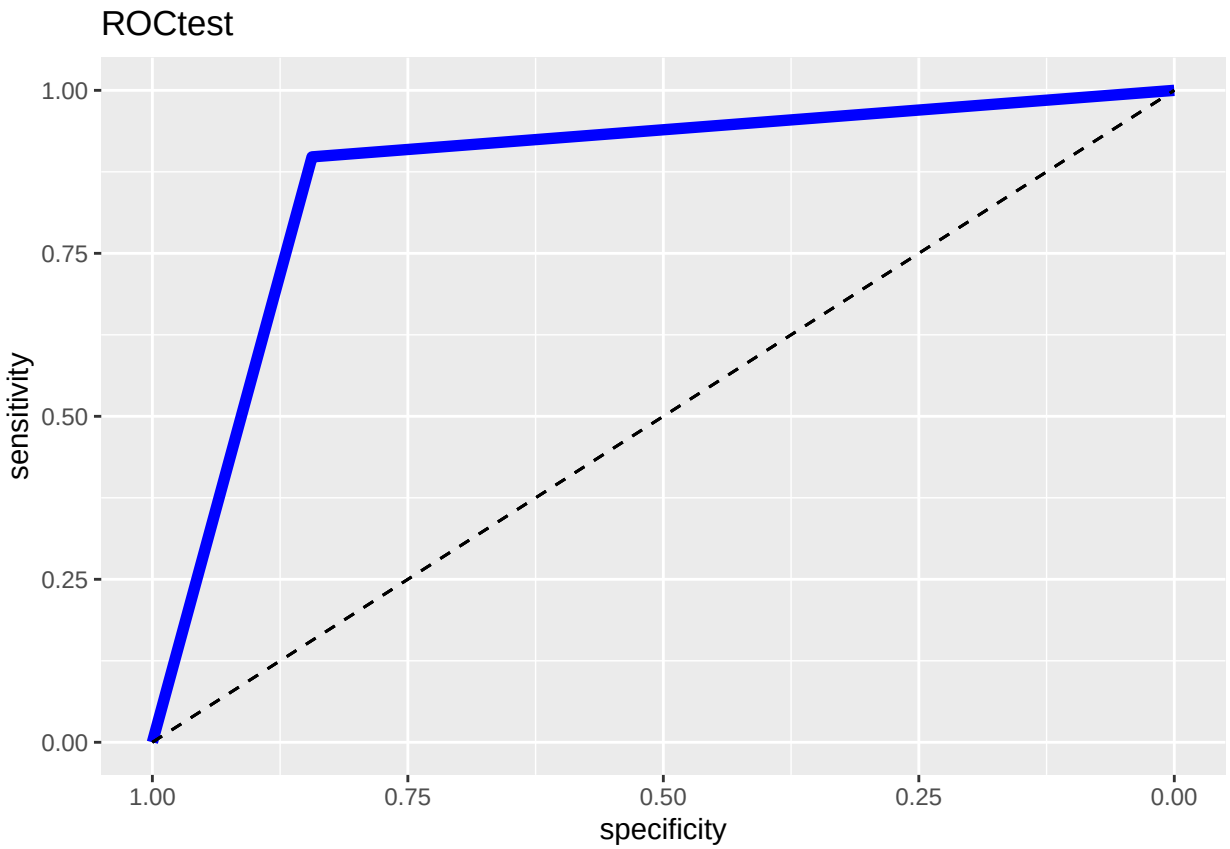
```r
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~Ranking + stlPerGameTeam + fg3mPerGameOpponent +
                   fg3mPerGameTeam + ortgTeamMisc + pctTOVOpponentMisc +
                   fg3aPerGameOpponent + winsTeam + stlPerGameOpponent +
                   fg2mPerGameTeam + pctFTPerGameTeam + fg2aPerGameTeam +
                   pctFGPerGameTeam, 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[[2]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[2]]
```

```
##        y_pred
## y_true   0   1
##      0  54  10
##      1  15 132
```

```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$ytest):max(datatest$y
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 + drtgTeamMisc + pctFGPerGameOpponent +
                   ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam,
                 data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc + drtgTeamMisc + pctFGPerGameOpponent +
##     ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam, data = datatrain)
##
## Prior probabilities of groups:
##        0        1
## 0.296063 0.703937
##
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.1365256  1.0603001    1.0151935         1.0128761        -0.7899802
## 1  0.4423745 -0.4194536   -0.3919078        -0.3915505         0.2773244
```

```
##   drtgTeamMisc pctFGPerGameOpponent ortgTeamMisc pctFG2PerGameOpponent
## 0   -0.7869574           -0.7695938    0.7837865            -0.7378330
## 1    0.2791195            0.2742821   -0.3178095             0.2418056
##   pctTrueShootingeTeamMisc pctEFGTeamMisc pctFG2PerGameTeam pctFGPerGameTeam
## 0                0.7229441      0.7378573         0.7051447        0.6671154
## 1               -0.3204302     -0.3204519        -0.3044230       -0.2957405
##
## Coefficients of linear discriminants:
##                                 LD1
## Ranking                  1.58905860
## winsTeam                -0.07820737
## nrtgTeamMisc            -2.34254959
## marginVictoryTeam        2.55762927
## pctEFGTeamOppMisc        0.66520506
## drtgTeamMisc            -0.30198226
## pctFGPerGameOpponent    -0.27593437
## ortgTeamMisc             0.07100852
## pctFG2PerGameOpponent   -0.16752019
## pctTrueShootingeTeamMisc -0.01623399
## pctEFGTeamMisc          -0.02826062
## pctFG2PerGameTeam       -0.08930459
## pctFGPerGameTeam         0.01451943
```

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

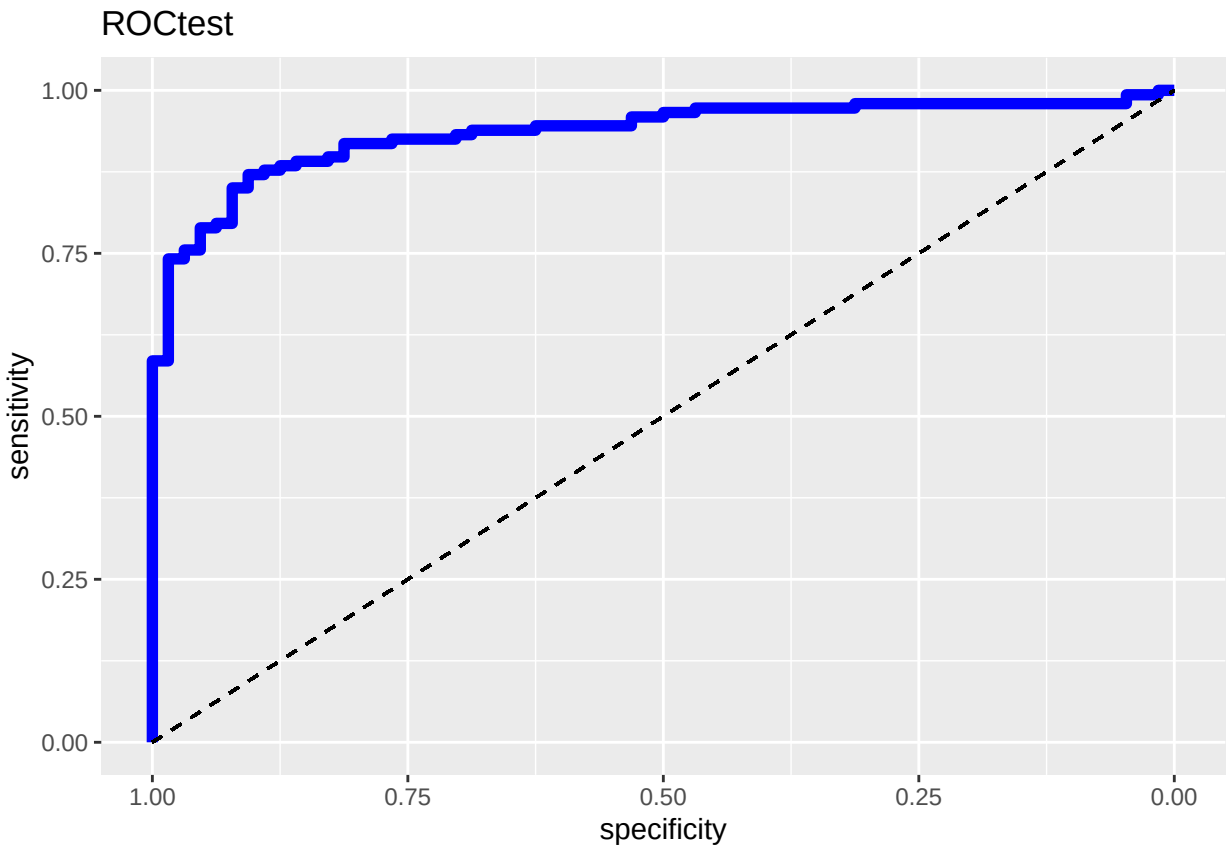
```
##       y_pred
## y_true   0   1
##      0  56   8
##      1  18 129
```

```
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 +
                    pctEFGTeamOppMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + pctFG2PerGameOpponent +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + pctFGPerGameTeam +
                    ptsPerGameOpponent + drbPerGameTeam +
                    pctFG3PerGameOpponent + astPerGameOpponent +
                    ageMeanMisc + blkPerGameOpponent,
                  data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[2]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[2]]
```

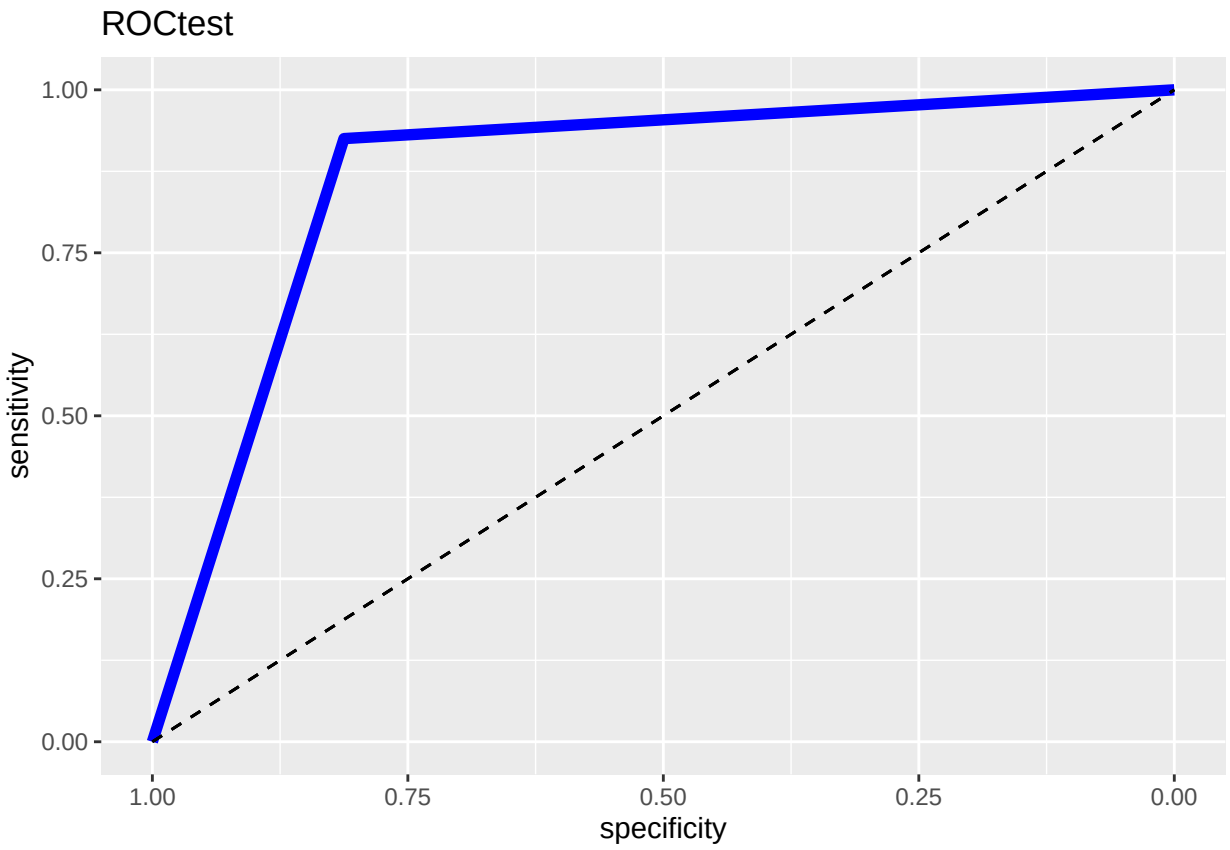
```
##        y_pred
## y_true   0   1
##      0  52  12
##      1  11 136
```

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

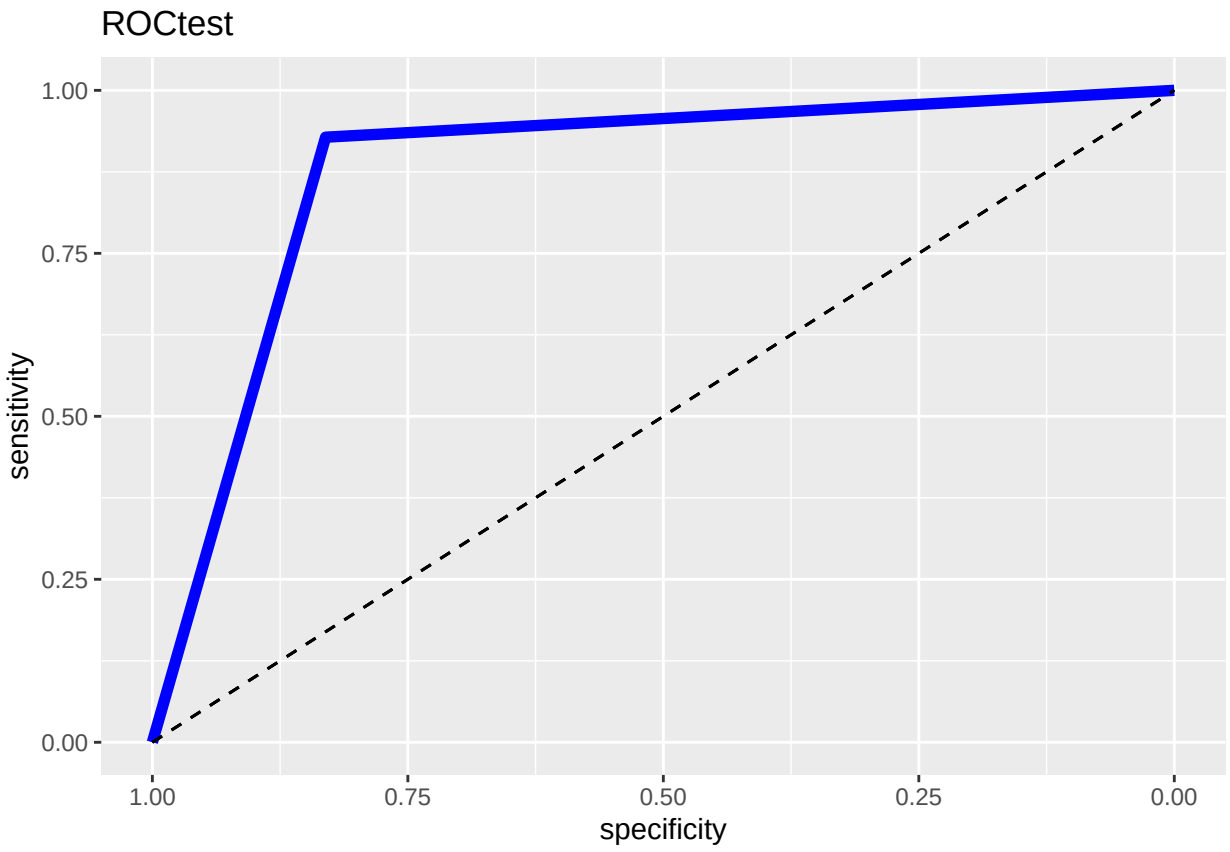
```r
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~Ranking + stlPerGameTeam + fg3mPerGameOpponent +
                   fg3mPerGameTeam + ortgTeamMisc + pctTOVOpponentMisc +
                   fg3aPerGameOpponent + winsTeam + stlPerGameOpponent +
                   fg2mPerGameTeam + pctFTPerGameTeam + fg2aPerGameTeam +
                   pctFGPerGameTeam, 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  49  10
##      1  11 142
```

```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$ytest):max(datatest$y
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 + drtgTeamMisc + pctFGPerGameOpponent +
                   ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam,
                 data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc + drtgTeamMisc + pctFGPerGameOpponent +
##     ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.3044164 0.6955836
##
## Group means:
##      Ranking   winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.1207305  1.0533436    1.0176262         1.0153070        -0.7774261
## 1  0.4312412 -0.4049765   -0.3751997        -0.3735137         0.2672065
```

```
##   drtgTeamMisc pctFGPerGameOpponent ortgTeamMisc pctFG2PerGameOpponent
## 0    -0.799232           -0.7665624    0.7792068            -0.7240927
## 1     0.297113            0.2664398   -0.2846144             0.2421448
##   pctTrueShootingeTeamMisc pctEFGTeamMisc pctFG2PerGameTeam pctFGPerGameTeam
## 0                 0.737290      0.7569746         0.7283674        0.7012115
## 1                -0.283533     -0.2980026        -0.2576451       -0.2585353
##
## Coefficients of linear discriminants:
##                                 LD1
## Ranking                  1.58123455
## winsTeam                -0.07683333
## nrtgTeamMisc            -0.73923883
## marginVictoryTeam        1.20572023
## pctEFGTeamOppMisc        0.48009984
## drtgTeamMisc             0.02582067
## pctFGPerGameOpponent    -0.25748457
## ortgTeamMisc             0.03803051
## pctFG2PerGameOpponent   -0.13859064
## pctTrueShootingeTeamMisc 0.02692953
## pctEFGTeamMisc          -0.25943550
## pctFG2PerGameTeam        0.03271698
## pctFGPerGameTeam        -0.01329083
```

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

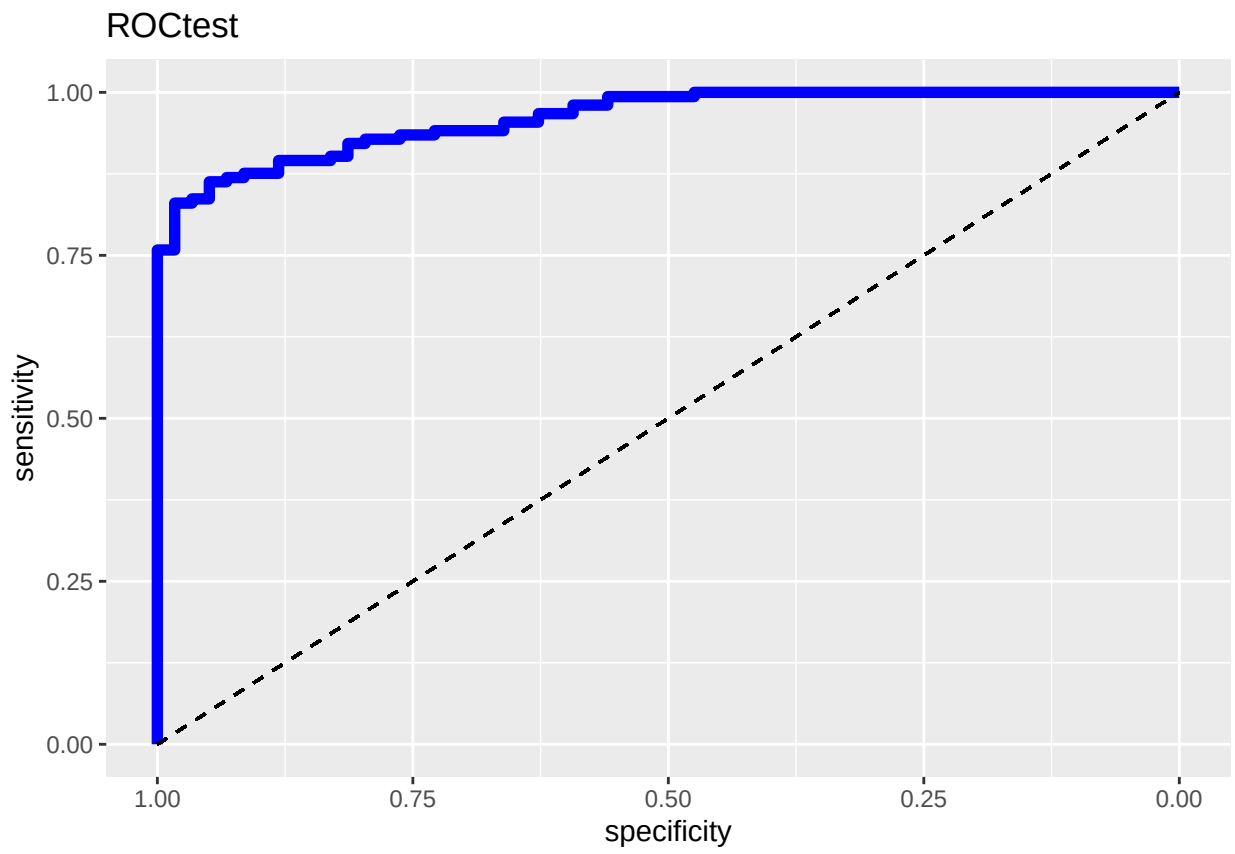
```
##       y_pred
## y_true   0   1
##      0  51   8
##      1  16 137
```

```
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 +
                    pctEFGTeamOppMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + pctFG2PerGameOpponent +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + pctFGPerGameTeam +
                    ptsPerGameOpponent + drbPerGameTeam +
                    pctFG3PerGameOpponent + astPerGameOpponent +
                    ageMeanMisc + blkPerGameOpponent,
                  data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[3]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[3]]
```

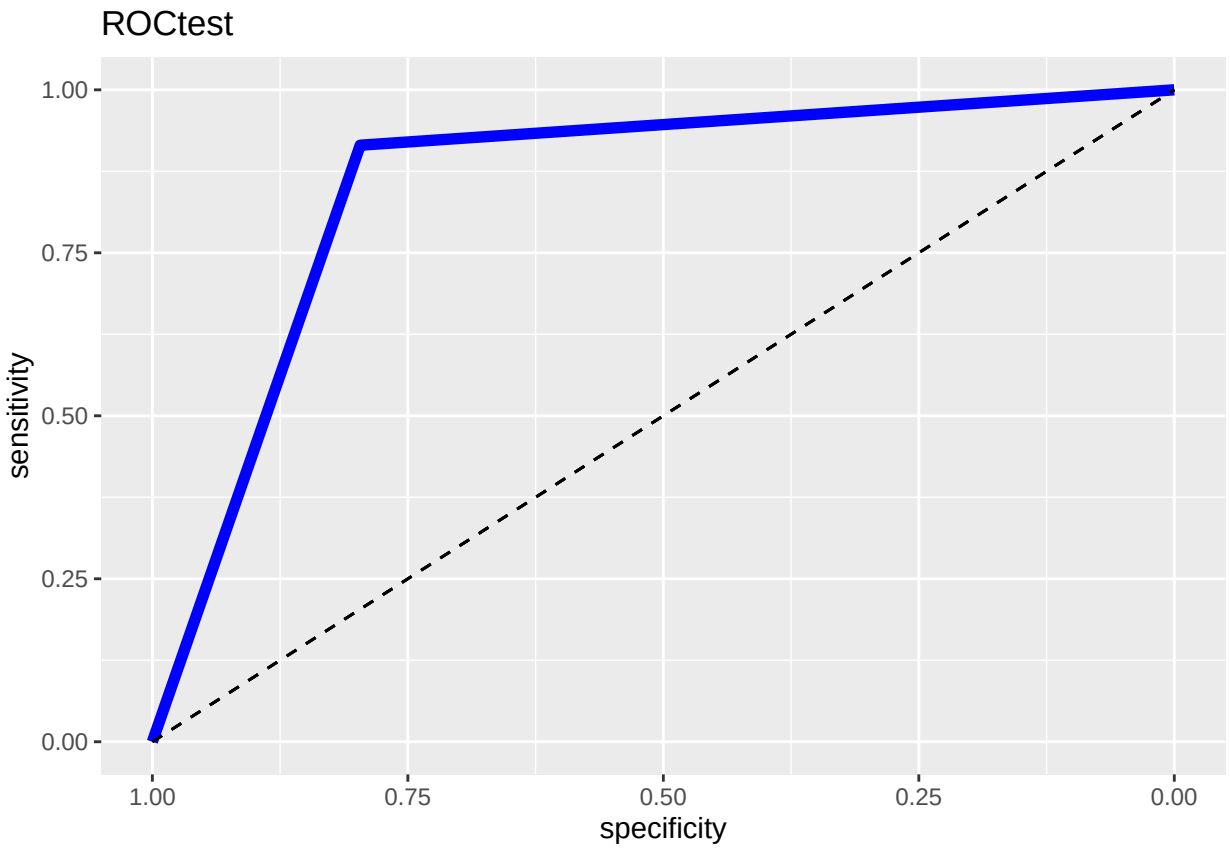
```
##        y_pred
## y_true   0   1
##      0  47  12
##      1  13 140
```

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

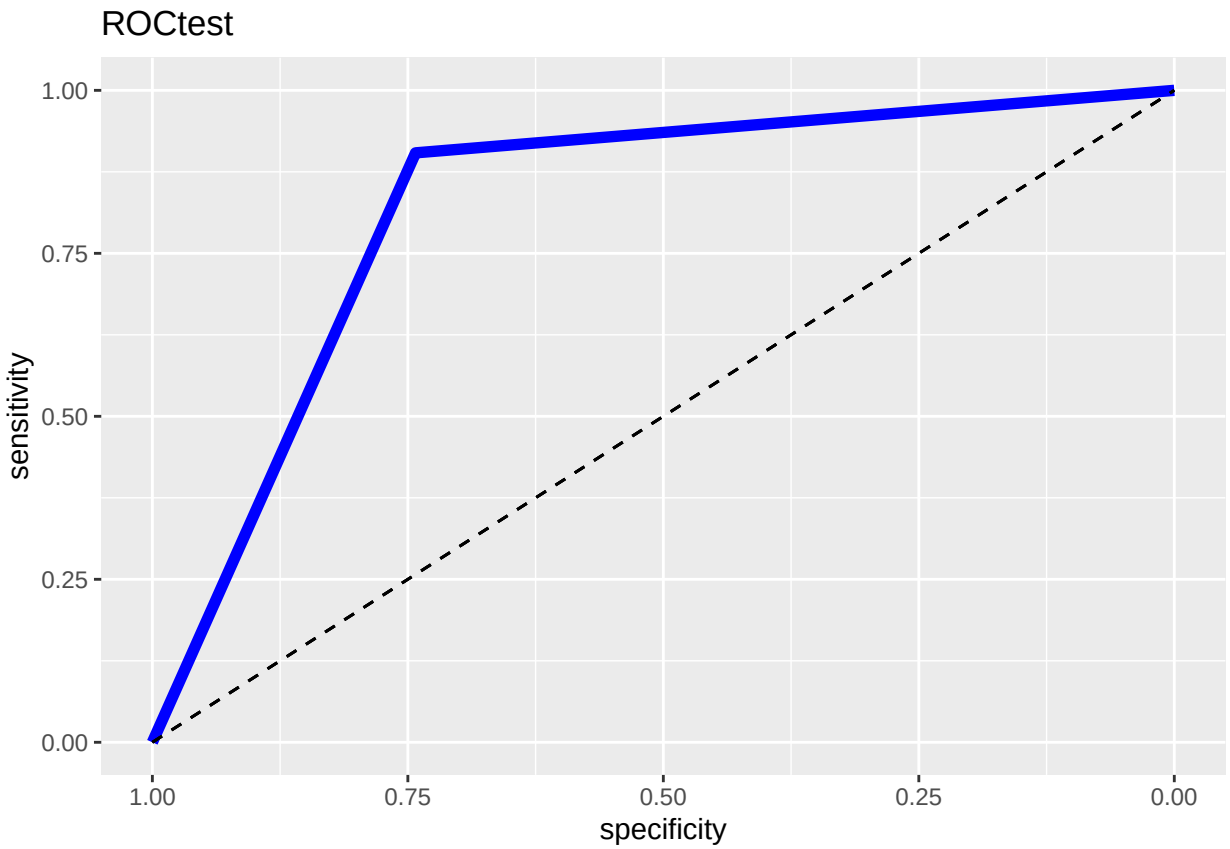
```r
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~Ranking + stlPerGameTeam + fg3mPerGameOpponent +
                   fg3mPerGameTeam + ortgTeamMisc + pctTOVOpponentMisc +
                   fg3aPerGameOpponent + winsTeam + stlPerGameOpponent +
                   fg2mPerGameTeam + pctFTPerGameTeam + fg2aPerGameTeam +
                   pctFGPerGameTeam, 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  49  17
##      1  14 132
```

```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$ytest):max(datatest$y
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 + drtgTeamMisc + pctFGPerGameOpponent +
                   ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam,
                 data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc + drtgTeamMisc + pctFGPerGameOpponent +
##     ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.2933754 0.7066246
##
## Group means:
##      Ranking  winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.1471810  1.087272    1.0603179         1.0562269        -0.8441365
## 1  0.4348507 -0.418309   -0.3885997        -0.3872321         0.2651396
```

```
##   drtgTeamMisc pctFGPerGameOpponent ortgTeamMisc pctFG2PerGameOpponent
## 0   -0.8510235           -0.8152217    0.7761714            -0.7749893
## 1    0.2797476            0.2607520   -0.3139240             0.2305070
##   pctTrueShootingeTeamMisc pctEFGTeamMisc pctFG2PerGameTeam pctFGPerGameTeam
## 0                0.6881784      0.7341607         0.6958845        0.6726524
## 1               -0.3105748     -0.3106578        -0.2807011       -0.2726783
##
## Coefficients of linear discriminants:
##                                 LD1
## Ranking                  1.59953134
## winsTeam                 0.03001404
## nrtgTeamMisc            -1.48194704
## marginVictoryTeam        2.01253953
## pctEFGTeamOppMisc        0.81868026
## drtgTeamMisc            -0.02662202
## pctFGPerGameOpponent    -0.27427387
## ortgTeamMisc            -0.16566650
## pctFG2PerGameOpponent   -0.33125531
## pctTrueShootingeTeamMisc 0.15329656
## pctEFGTeamMisc          -0.32283454
## pctFG2PerGameTeam        0.12286744
## pctFGPerGameTeam        -0.11562911
```

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

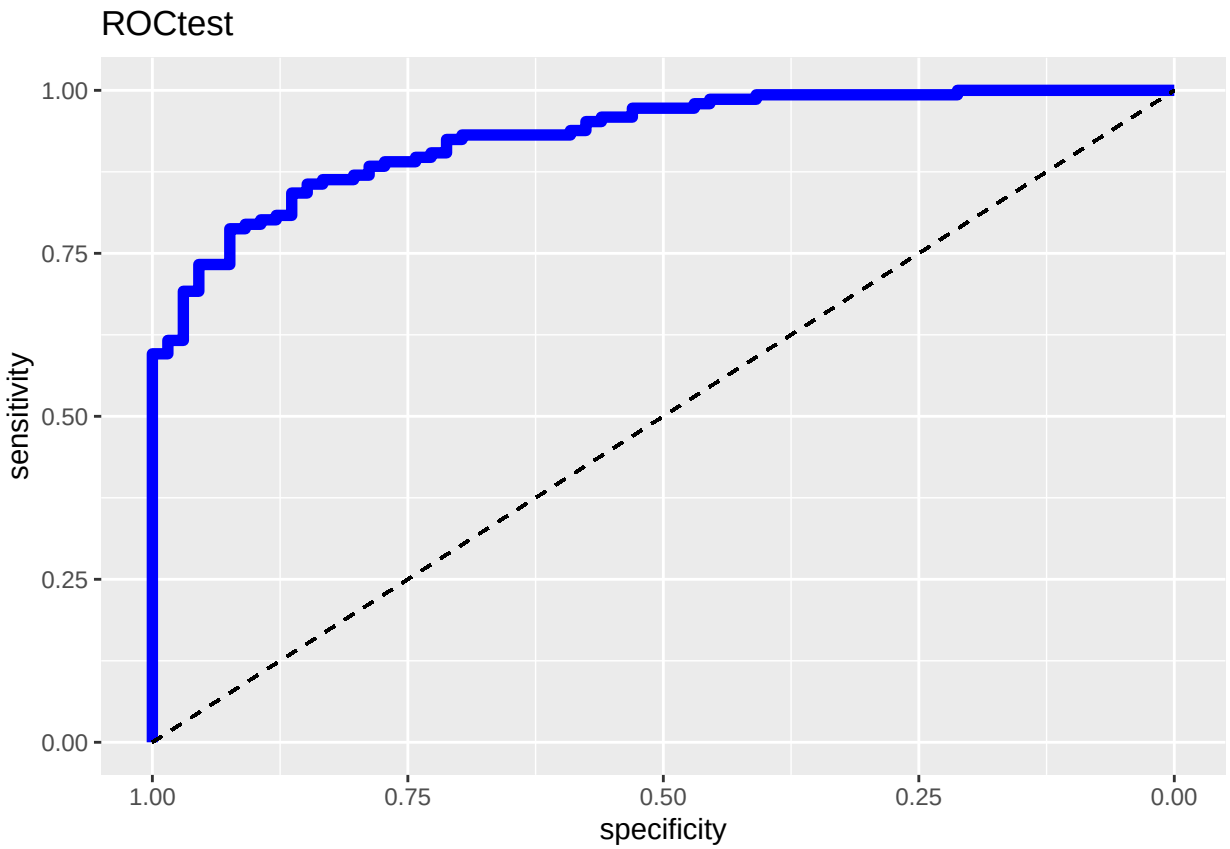
```
##       y_pred
## y_true   0   1
##      0  53  13
##      1  20 126
```

```
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 +
                    pctEFGTeamOppMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctTrueShootingeTeamMisc + pctFG2PerGameOpponent +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + pctFGPerGameTeam +
                    ptsPerGameOpponent + drbPerGameTeam +
                    pctFG3PerGameOpponent + astPerGameOpponent +
                    ageMeanMisc + blkPerGameOpponent,
                  data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[4]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[4]]
```

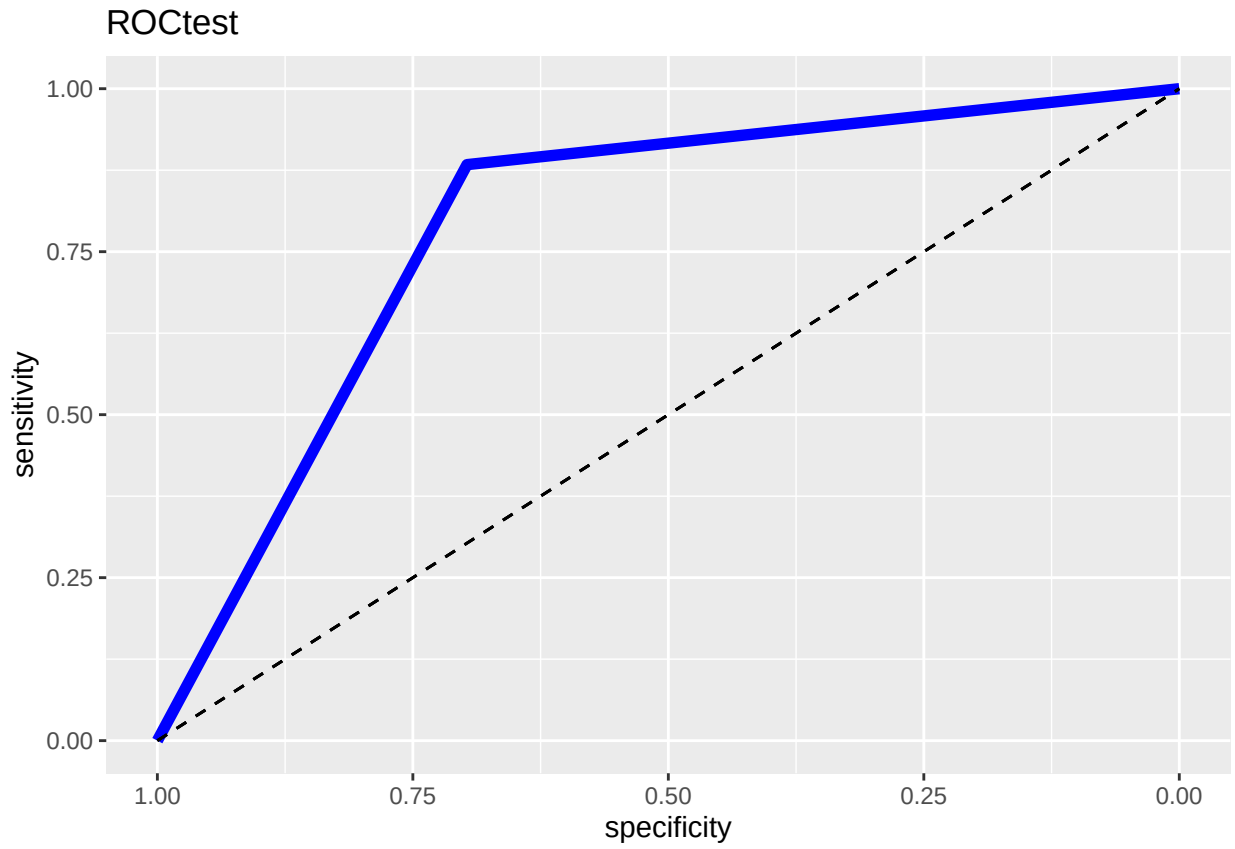
```
##        y_pred
## y_true   0   1
##      0  46  20
##      1  17 129
```

```
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] 0.1146383
```

```
Accuracyglm/4
```

```
## [1] 0.8853617
```

```
Precisionglm/4
```

```
## [1] 0.807378
```

```
Recallglm/4
```

```
## [1] 0.8105199
```

```
F1glm/4
```

```
## [1] 0.8084898
```

```
AUCglm/4
```

```
## [1] 0.8639303
```

```
ConfusMatglm
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0  52  11
##      1   9 139
##
## [[2]]
##       y_pred
## y_true   0   1
##      0  54  10
##      1  15 132
##
## [[3]]
##       y_pred
## y_true   0   1
##      0  49  10
##      1  11 142
##
## [[4]]
##       y_pred
## y_true   0   1
##      0  49  17
##      1  14 132
```

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

```
## [1] 1.055536
```

```
Accuracylda/4
```

```
## [1] 0.875911
```

```
Precisionlda/4
```

```
## [1] 0.7624871
```

```
Recalllda/4
```

```
## [1] 0.849895
```

```
F1lda/4
```

```
## [1] 0.8036193
```

```
AUClda/4
```

```
## [1] 0.8684665
```

```
ConfusMatlda
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0  54   9
##      1  13 135
##
## [[2]]
##       y_pred
## y_true   0   1
##      0  56   8
##      1  18 129
##
## [[3]]
##       y_pred
## y_true   0   1
##      0  51   8
##      1  16 137
##
## [[4]]
##       y_pred
## y_true   0   1
##      0  53  13
##      1  20 126
```

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

```
## [1] 0.1335397
```

```
Accuracyknn/4
```

```
## [1] 0.8664603
```

```
Precisionknn/4
```

```
## [1] 0.7864766
```

```
Recallknn/4
```

```
## [1] 0.7590596
```

```
F1knn/4
```

```
## [1] 0.7721646
```

```
AUCknn/4
```

```
## [1] 0.8357098
```

```
ConfusMatknn
```

```
## [[1]]
##        y_pred
## y_true   0   1
##      0  46  17
##      1  11 137
##
## [[2]]
##        y_pred
## y_true   0   1
##      0  52  12
##      1  11 136
##
## [[3]]
##        y_pred
## y_true   0   1
##      0  47  12
##      1  13 140
##
## [[4]]
##        y_pred
## y_true   0   1
##      0  46  20
##      1  17 129
```

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

##Logistic Regression
model.glm <- glm(ytrain~Ranking + stlPerGameTeam + fg3mPerGameOpponent +
                   fg3mPerGameTeam + ortgTeamMisc + pctTOVOpponentMisc +
                   fg3aPerGameOpponent + winsTeam + stlPerGameOpponent +
                   fg2mPerGameTeam + pctFTPerGameTeam + fg2aPerGameTeam +
                   pctFGPerGameTeam, 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  0  1  1  1  1  1  0  1  1  1  0  0  0  1  0  0  1  1  1  0  1  1  1  1  1
## 27 28 29 30
##  1  0  0  1
```

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

```
## [1] "Boston Celtics"
## [1] "Denver Nuggets"
## [1] "Indiana Pacers"
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Miami Heat"
## [1] "Milwaukee Bucks"
## [1] "Oklahoma City Thunder"
## [1] "Toronto Raptors"
## [1] "Utah Jazz"
```

```
##Logistic Regression Model suggests that the Boston Celtics, Denver Nuggets,
##Indiana Pacers, LA Clippers, Los Angeles Lakers, Miami Heat, Milwaukee Bucks,
##Oklahoma City Thunder, Toronto Raptors and Utah Jazz are Second Round level teams

##Discriminant Analysis
##Linear Discriminant
model.lda <- lda(ytrain~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
```

```
                    pctEFGTeamOppMisc + drtgTeamMisc + pctFGPerGameOpponent +
                    ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
                    pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam,
                  data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + nrtgTeamMisc + marginVictoryTeam +
##     pctEFGTeamOppMisc + drtgTeamMisc + pctFGPerGameOpponent +
##     ortgTeamMisc + pctFG2PerGameOpponent + pctTrueShootingeTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam, data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.2718676 0.7281324
##
## Group means:
##      Ranking  winsTeam nrtgTeamMisc marginVictoryTeam pctEFGTeamOppMisc
## 0 -1.1438427  1.067327    1.0279128          1.026668        -0.8284682
## 1  0.4270841 -0.398515   -0.3837986         -0.383334         0.3093307
##   drtgTeamMisc pctFGPerGameOpponent ortgTeamMisc pctFG2PerGameOpponent
## 0   -0.8298686           -0.8155122    0.7650806            -0.7588601
## 1    0.3098535            0.3044932   -0.2856632             0.2833406
##   pctTrueShootingeTeamMisc pctEFGTeamMisc pctFG2PerGameTeam pctFGPerGameTeam
## 0                0.7649825      0.7608188         0.7120504        0.6851752
## 1               -0.2856266     -0.2840719        -0.2658630       -0.2558284
##
## Coefficients of linear discriminants:
##                                 LD1
## Ranking                  1.61175268
## winsTeam                 0.26880294
## nrtgTeamMisc            -1.83267634
## marginVictoryTeam        1.51701050
## pctEFGTeamOppMisc        0.74006697
## drtgTeamMisc            -0.43457042
## pctFGPerGameOpponent    -0.16597727
## ortgTeamMisc             0.39658639
## pctFG2PerGameOpponent   -0.29307724
## pctTrueShootingeTeamMisc -0.19378502
## pctEFGTeamMisc          -0.00796552
## pctFG2PerGameTeam       -0.07750641
## pctFGPerGameTeam         0.05286904
```

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

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

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

```
  }
}
```

```
## [1] "Boston Celtics"
## [1] "Denver Nuggets"
## [1] "Indiana Pacers"
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Miami Heat"
## [1] "Milwaukee Bucks"
## [1] "Oklahoma City Thunder"
## [1] "Toronto Raptors"
## [1] "Utah Jazz"
```

```
##Discriminant Analysis Model suggests that the Boston Celtics, Denver Nuggets,
##Indiana Pacers, LA Clippers, Los Angeles Lakers, Miami Heat, Milwaukee Bucks,
##Oklahoma City Thunder, Toronto Raptors and Utah Jazz are Second Round level teams

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

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

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

```
## [1] "Boston Celtics"
## [1] "Dallas Mavericks"
## [1] "Indiana Pacers"
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Miami Heat"
## [1] "Milwaukee Bucks"
## [1] "Oklahoma City Thunder"
## [1] "Toronto Raptors"
## [1] "Utah Jazz"
```

```
##K Nearest Neighbours suggests that the Boston Celtics, Dallas Mavericks,
##Denver Nuggets, Indiana Pacers, LA Clippers, Los Angeles Lakers,
##Miami Heat, Milwaukee Bucks, Oklahoma City Thunder, Toronto Raptors and
##Utah Jazz are Second Round level teams
```