NBAAnalysisFinalsClassificationModel.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)

##Finalist Feature Selection
ytrain <- ceiling((MasterPerGame$finish-1)/5)
xtrain <- MasterPerGame[,-3]
datatrain <- cbind(ytrain, xtrain)

##Generalized Linear Model Feature Selection
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
```

```
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), meth
```

```
## 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   0.9184 0.4034   0.004471 0.04745
##          6   0.9208 0.3856   0.006958 0.06566
##          7   0.9303 0.4752   0.002475 0.02513        *
##          8   0.9196 0.3784   0.013908 0.12762
##          9   0.9184 0.3665   0.013027 0.11298
##         10   0.9267 0.4394   0.016593 0.10660
##         11   0.9267 0.4479   0.017488 0.11663
##         12   0.9303 0.5094   0.018311 0.09834
##         13   0.9267 0.4688   0.016217 0.15405
##         14   0.9232 0.4524   0.020573 0.11169
##         15   0.9196 0.4208   0.019335 0.11483
##         16   0.9196 0.4227   0.019291 0.10162
##         17   0.9184 0.4193   0.020574 0.09116
##         18   0.9125 0.3857   0.019940 0.10436
##         19   0.9208 0.4512   0.026625 0.15575
##         20   0.9255 0.4811   0.019537 0.12360
##         21   0.9232 0.4635   0.017095 0.10762
##         22   0.9184 0.4406   0.020975 0.11828
##         23   0.9208 0.4561   0.020586 0.12582
##         24   0.9196 0.4442   0.018624 0.11115
##         25   0.9232 0.4601   0.022036 0.12660
##         62   0.9054 0.4300   0.013972 0.06773
##
## The top 5 variables (out of 7):
##    Ranking, tovPerGameOpponent, blkPerGameOpponent, fg3mPerGameTeam, ptsPerGameOpponent
```

```
model.glm$optVariables
```

```
## [1] "Ranking"            "tovPerGameOpponent" "blkPerGameOpponent"
## [4] "fg3mPerGameTeam"    "ptsPerGameOpponent" "ftmPerGameOpponent"
## [7] "pctTOVOpponentMisc"
```

```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   0.9137 0.05709   0.002232 0.07410
##          6   0.9208 0.20595   0.004602 0.04290
##          7   0.9196 0.18675   0.005464 0.04420
##          8   0.9173 0.14375   0.006006 0.07857
##          9   0.9184 0.14836   0.007970 0.08693
##         10   0.9137 0.11807   0.005785 0.05296
##         11   0.9137 0.13361   0.005785 0.07702
##         12   0.9125 0.13022   0.006001 0.07602
##         13   0.9137 0.13341   0.007044 0.07369
##         14   0.9161 0.13924   0.008199 0.10905
##         15   0.9161 0.15837   0.007134 0.06798
##         16   0.9149 0.15441   0.006774 0.06248
##         17   0.9161 0.17486   0.009043 0.08548
##         18   0.9172 0.20667   0.008333 0.09265
##         19   0.9196 0.24457   0.008647 0.09488
##         20   0.9232 0.25621   0.007049 0.08565        *
##         21   0.9220 0.25166   0.006202 0.08089
##         22   0.9208 0.24796   0.007060 0.09013
##         23   0.9196 0.24429   0.008566 0.09211
##         24   0.9173 0.21877   0.004594 0.07722
##         25   0.9196 0.24007   0.004019 0.08302
##         62   0.9078 0.21773   0.009705 0.18331
##
## The top 5 variables (out of 20):
##    Ranking, winsTeam, marginVictoryTeam, nrtgTeamMisc, pctEFGTeamMisc
```

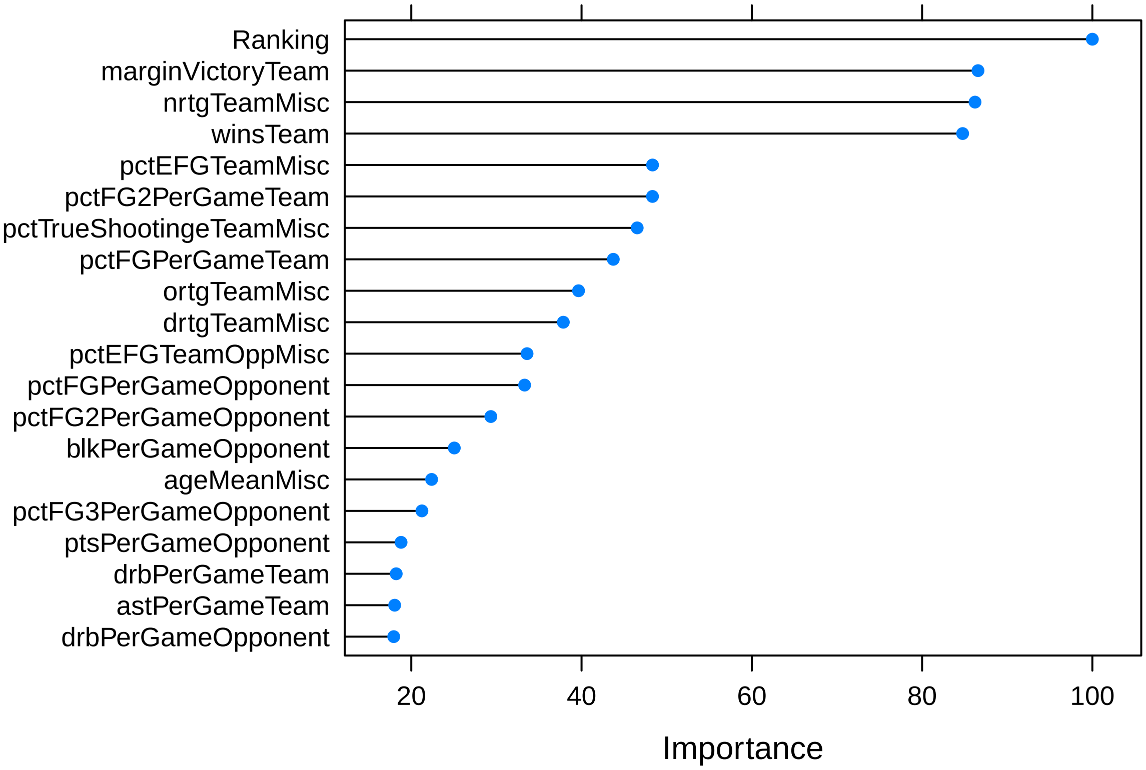
```r
model.lda$optVariables
```

```
##  [1] "Ranking"                  "winsTeam"
##  [3] "marginVictoryTeam"        "nrtgTeamMisc"
##  [5] "pctEFGTeamMisc"           "pctFG2PerGameTeam"
##  [7] "drtgTeamMisc"             "pctTrueShootingeTeamMisc"
##  [9] "pctFGPerGameTeam"         "pctFGPerGameOpponent"
## [11] "ortgTeamMisc"             "pctEFGTeamOppMisc"
## [13] "pctFG2PerGameOpponent"    "blkPerGameOpponent"
## [15] "ageMeanMisc"              "ptsPerGameOpponent"
## [17] "drbPerGameTeam"           "astPerGameOpponent"
## [19] "drbPerGameOpponent"       "ptsPerGameTeam"
```

```r
##KNN Feature Selection
##Note we cannot apply rfe methods to KNN
##thus, we shall take variables with importance above 20%
set.seed(2)
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
## marginVictoryTeam            86.57
## nrtgTeamMisc                 86.22
## winsTeam                     84.76
## pctEFGTeamMisc               48.34
## pctFG2PerGameTeam            48.33
## pctTrueShootingeTeamMisc     46.54
## pctFGPerGameTeam             43.73
## ortgTeamMisc                 39.65
## drtgTeamMisc                 37.86
## pctEFGTeamOppMisc            33.60
## pctFGPerGameOpponent         33.31
## pctFG2PerGameOpponent        29.34
## blkPerGameOpponent           25.05
## ageMeanMisc                  22.39
## pctFG3PerGameOpponent        21.25
## ptsPerGameOpponent           18.80
## drbPerGameTeam               18.23
## astPerGameTeam               18.05
## drbPerGameOpponent           17.94
```

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



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

##Finalist 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]))-1)/5)
```

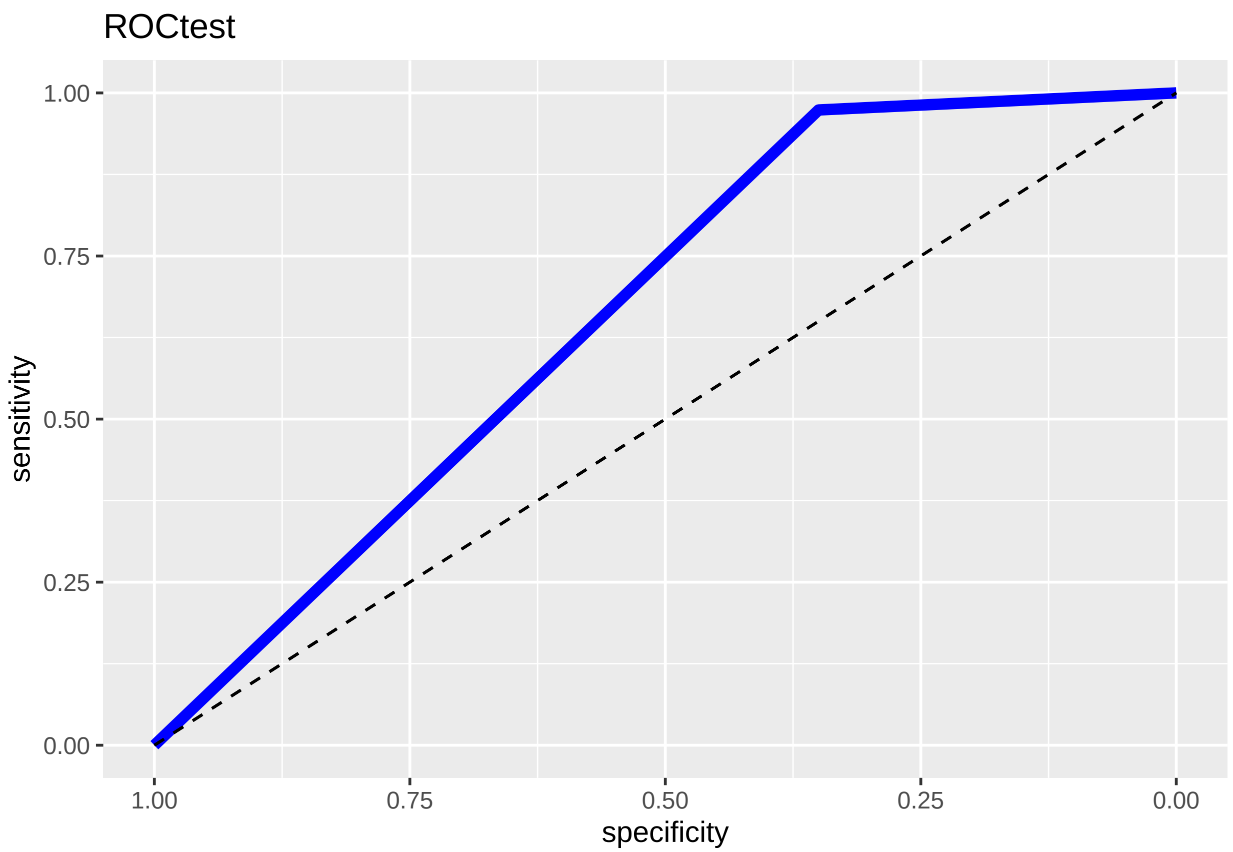
```r
xtrain <- rbind(PerGameFold2[,-3],PerGameFold3[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold1[,3]-1)/5)
xtest <- cbind(PerGameFold1[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~Ranking + tovPerGameOpponent + blkPerGameOpponent +
                   fg3mPerGameTeam + ptsPerGameOpponent +
                   ftmPerGameOpponent + pctTOVOpponentMisc,
                 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)), f
ConfusMatglm[[1]]
```

```
##       y_pred
## y_true   0   1
##      0   7  13
##      1   5 186
```

```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 + marginVictoryTeam + nrtgTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc +
                   pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                   pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
                   pctFG2PerGameOpponent + ageMeanMisc + blkPerGameOpponent +
                   ptsPerGameOpponent + drbPerGameTeam + astPerGameOpponent +
                   drbPerGameOpponent + ptsPerGameTeam, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc + pctTrueShootingeTeamMisc +
##     pctFGPerGameTeam + pctFGPerGameOpponent + ortgTeamMisc +
##     pctEFGTeamOppMisc + pctFG2PerGameOpponent + ageMeanMisc +
##     blkPerGameOpponent + ptsPerGameOpponent + drbPerGameTeam +
##     astPerGameOpponent + drbPerGameOpponent + ptsPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.1055118 0.8944882
```

```
##
## Group means:
##      Ranking   winsTeam marginVictoryTeam nrtgTeamMisc pctEFGTeamMisc
## 0 -1.3063775  1.2724448         1.3036583    1.3098406     0.84608484
## 1  0.1028363 -0.1118254        -0.1167422   -0.1171991    -0.08185865
##   pctFG2PerGameTeam drtgTeamMisc pctTrueShootingeTeamMisc pctFGPerGameTeam
## 0        0.93428146  -1.14695377               0.76111306       0.87268885
## 1       -0.08640088   0.09214476              -0.08265782      -0.08500882
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctFG2PerGameOpponent
## 0          -1.00772693   0.91174324        -1.0155478           -0.96528258
## 1           0.06130017  -0.09154708         0.0580346            0.04069203
##   ageMeanMisc blkPerGameOpponent ptsPerGameOpponent drbPerGameTeam
## 0  0.75182719         -0.7948428        -0.90959161     0.66161065
## 1 -0.07302421          0.1043887         0.05394604    -0.09695948
##   astPerGameOpponent drbPerGameOpponent ptsPerGameTeam
## 0        -0.87148644         -0.7596437     0.51781218
## 1         0.03832628          0.0438407    -0.07630413
##
## Coefficients of linear discriminants:
##                                  LD1
## Ranking                   0.74212313
## winsTeam                  1.16444713
## marginVictoryTeam         3.61183011
## nrtgTeamMisc             -4.56352752
## pctEFGTeamMisc            0.04658619
## pctFG2PerGameTeam        -0.91721187
## drtgTeamMisc              0.46698458
## pctTrueShootingeTeamMisc  0.73201597
## pctFGPerGameTeam          0.24397717
## pctFGPerGameOpponent      0.05165129
## ortgTeamMisc              0.07047534
## pctEFGTeamOppMisc         0.42396147
## pctFG2PerGameOpponent    -0.46729497
## ageMeanMisc              -0.14229963
## blkPerGameOpponent        0.23668906
## ptsPerGameOpponent       -0.36642471
## drbPerGameTeam            0.10687696
## astPerGameOpponent        0.09996681
## drbPerGameOpponent        0.41085987
## ptsPerGameTeam           -0.19344100
```

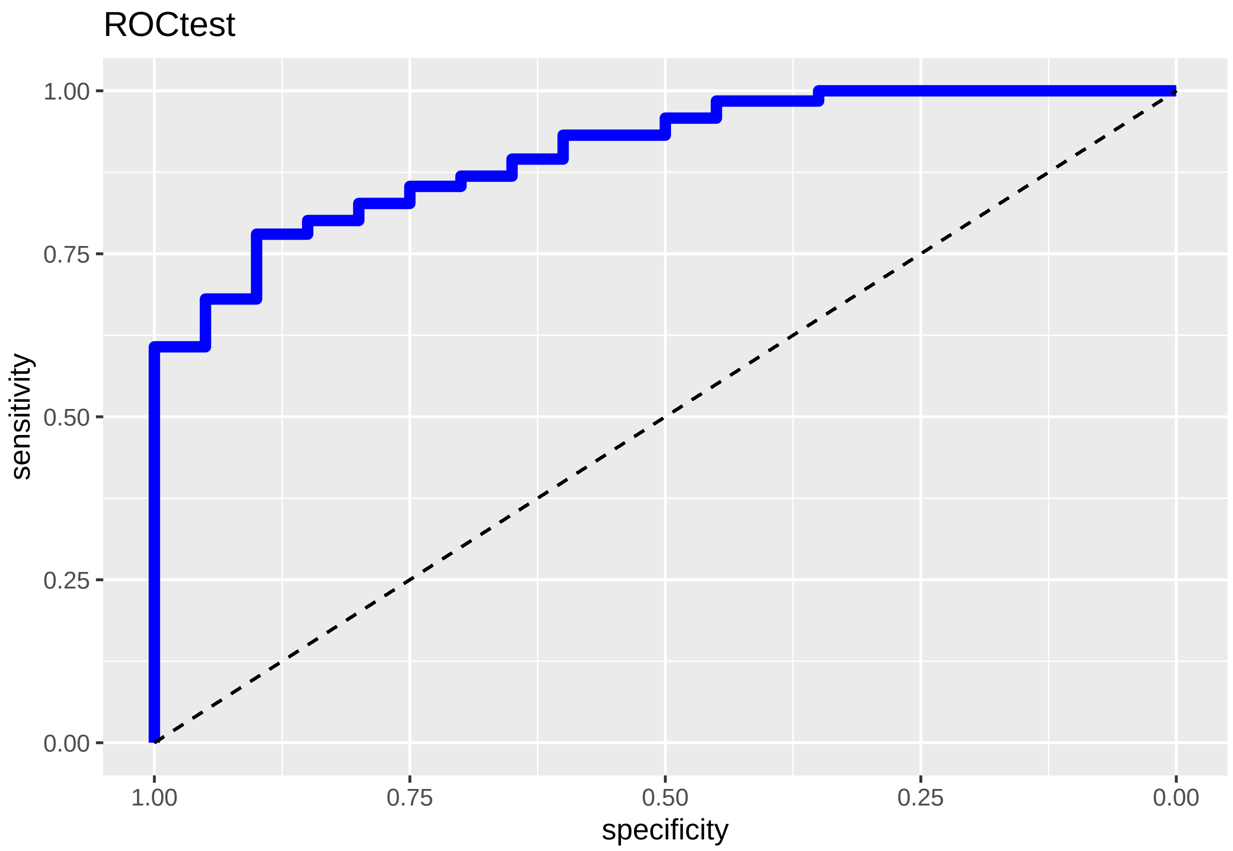
```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[1]] <-ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[1]]
```

```
##       y_pred
## y_true   0   1
##      0   7  13
##      1   3 188
```

```
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, 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 + marginVictoryTeam + nrtgTeamMisc +
                    winsTeam + pctEFGTeamMisc + pctFG2PerGameTeam +
                    pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                    ortgTeamMisc + drtgTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameOpponent +
                    blkPerGameOpponent + ageMeanMisc +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
```

```
ConfusMatknn[[1]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[1]]
```

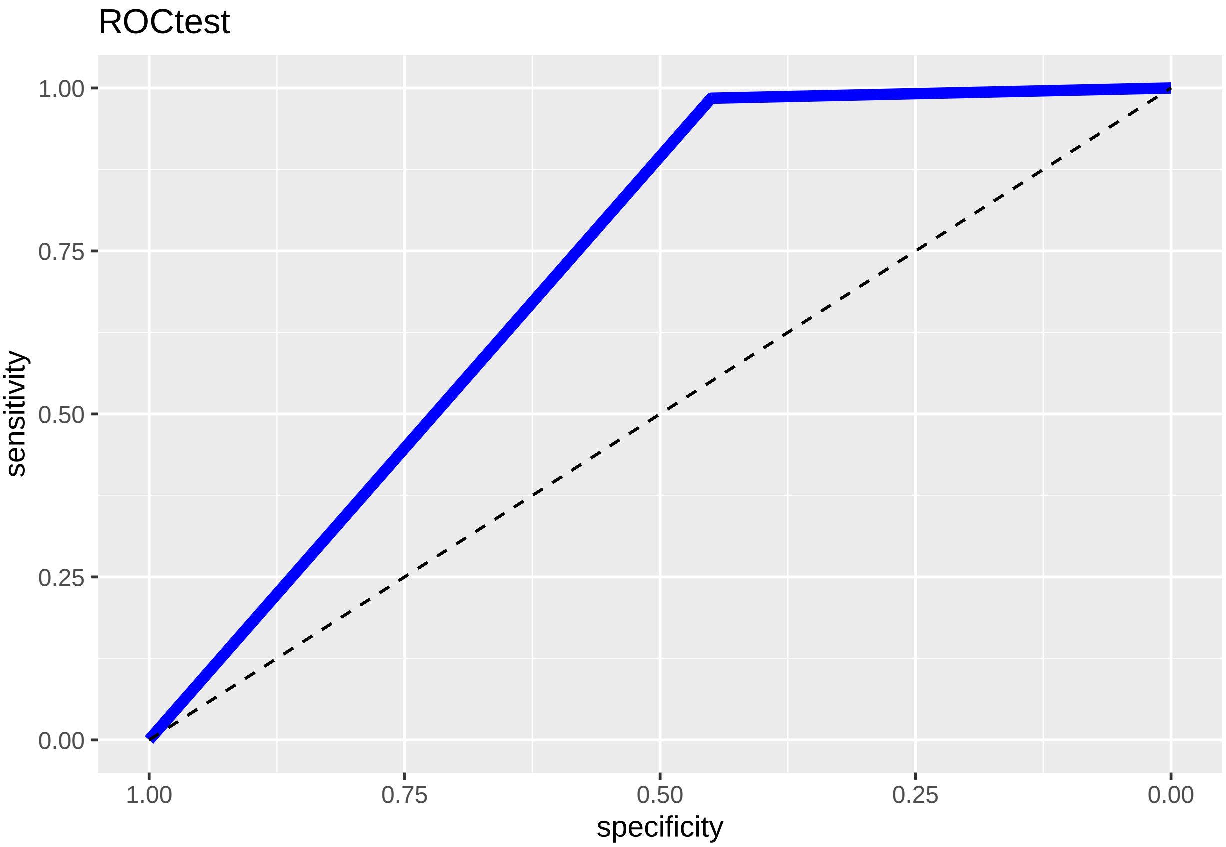
```
##       y_pred
## y_true   0   1
##      0   9  11
##      1   3 188
```

```
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()
```



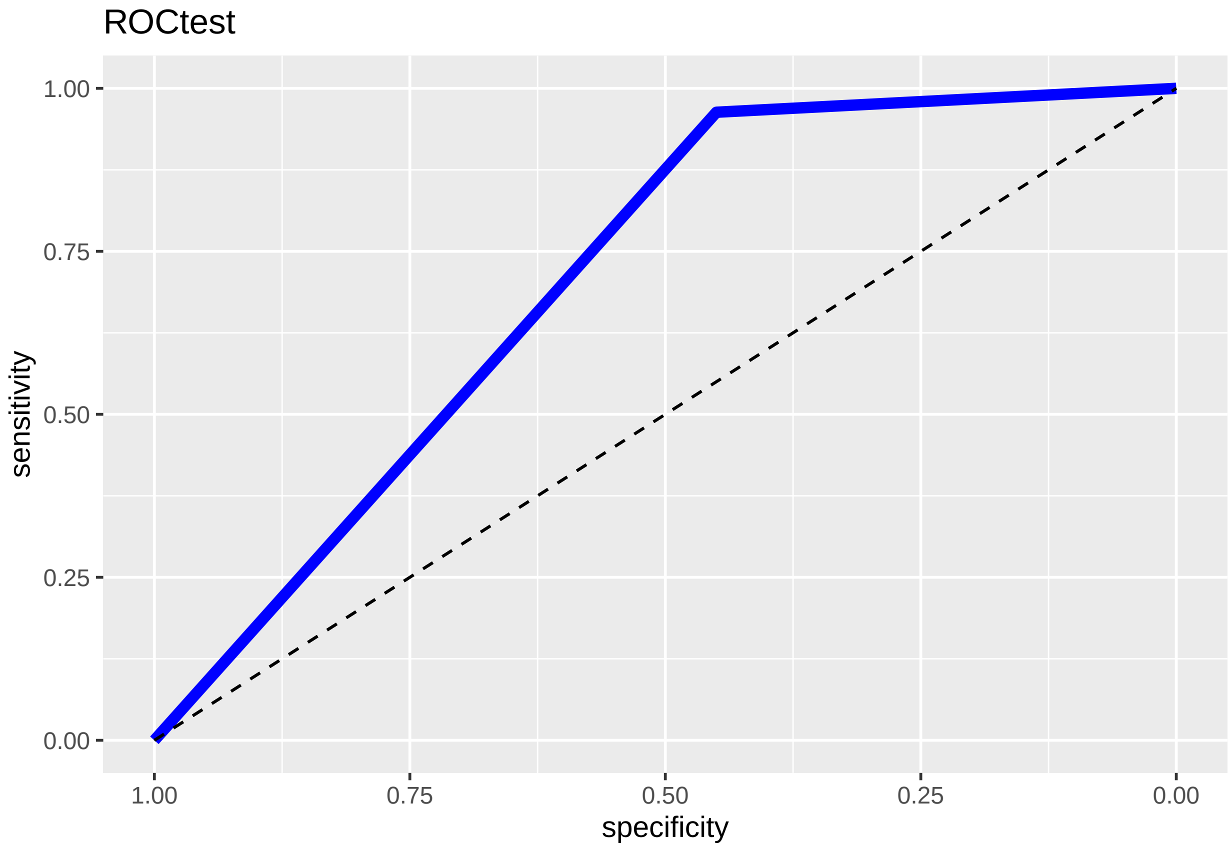
```r
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

##2nd fold = validation set
ytrain <- ceiling((rbind(cbind(PerGameFold1[,3]),cbind(PerGameFold3[,3]),cbind(PerGameFold4[,3]))-1)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold3[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold2[,3]-1)/5)
xtest <- cbind(PerGameFold2[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~Ranking + tovPerGameOpponent + blkPerGameOpponent +
                   fg3mPerGameTeam + ptsPerGameOpponent +
                   ftmPerGameOpponent + pctTOVOpponentMisc,
                 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   9  11
##      1   7 184
```

```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 + marginVictoryTeam + nrtgTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc +
                   pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                   pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
                   pctFG2PerGameOpponent + ageMeanMisc + blkPerGameOpponent +
                   ptsPerGameOpponent + drbPerGameTeam + astPerGameOpponent +
                   drbPerGameOpponent + ptsPerGameTeam, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc + pctTrueShootingeTeamMisc +
##     pctFGPerGameTeam + pctFGPerGameOpponent + ortgTeamMisc +
##     pctEFGTeamOppMisc + pctFG2PerGameOpponent + ageMeanMisc +
##     blkPerGameOpponent + ptsPerGameOpponent + drbPerGameTeam +
##     astPerGameOpponent + drbPerGameOpponent + ptsPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.1055118 0.8944882
```

```
##
## Group means:
##     Ranking   winsTeam marginVictoryTeam nrtgTeamMisc pctEFGTeamMisc
## 0 -1.329940  1.2650967          1.264661    1.2655652      0.9958071
## 1  0.128839 -0.1283817         -0.122068   -0.1216888     -0.1254294
##   pctFG2PerGameTeam drtgTeamMisc pctTrueShootingeTeamMisc pctFGPerGameTeam
## 0         1.0588070  -1.05742955                0.9330401        0.9928517
## 1        -0.1310738   0.08391939               -0.1229445       -0.1290482
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctFG2PerGameOpponent
## 0          -0.94895236    0.9402085       -0.96565895           -0.92338352
## 1           0.07306386   -0.1015897        0.07068112            0.05500213
##   ageMeanMisc blkPerGameOpponent ptsPerGameOpponent drbPerGameTeam
## 0  0.71388362        -0.76925196        -0.81636038     0.63314244
## 1 -0.06966568         0.06294693         0.06249015    -0.05112728
##   astPerGameOpponent drbPerGameOpponent ptsPerGameTeam
## 0        -0.81512859        -0.78561643     0.61613578
## 1         0.06303002         0.07379708    -0.06686562
##
## Coefficients of linear discriminants:
##                                  LD1
## Ranking                   1.07262768
## winsTeam                  0.81714892
## marginVictoryTeam         1.03299515
## nrtgTeamMisc             -0.67136463
## pctEFGTeamMisc           -0.38161309
## pctFG2PerGameTeam        -0.63195336
## drtgTeamMisc              0.64331705
## pctTrueShootingeTeamMisc  0.57214340
## pctFGPerGameTeam          0.20328250
## pctFGPerGameOpponent     -0.41736562
## ortgTeamMisc              0.20666924
## pctEFGTeamOppMisc         0.41385703
## pctFG2PerGameOpponent    -0.06340537
## ageMeanMisc              -0.09229911
## blkPerGameOpponent        0.14935059
## ptsPerGameOpponent        0.04480266
## drbPerGameTeam            0.15559103
## astPerGameOpponent        0.15481657
## drbPerGameOpponent        0.47454052
## ptsPerGameTeam           -0.69912232
```

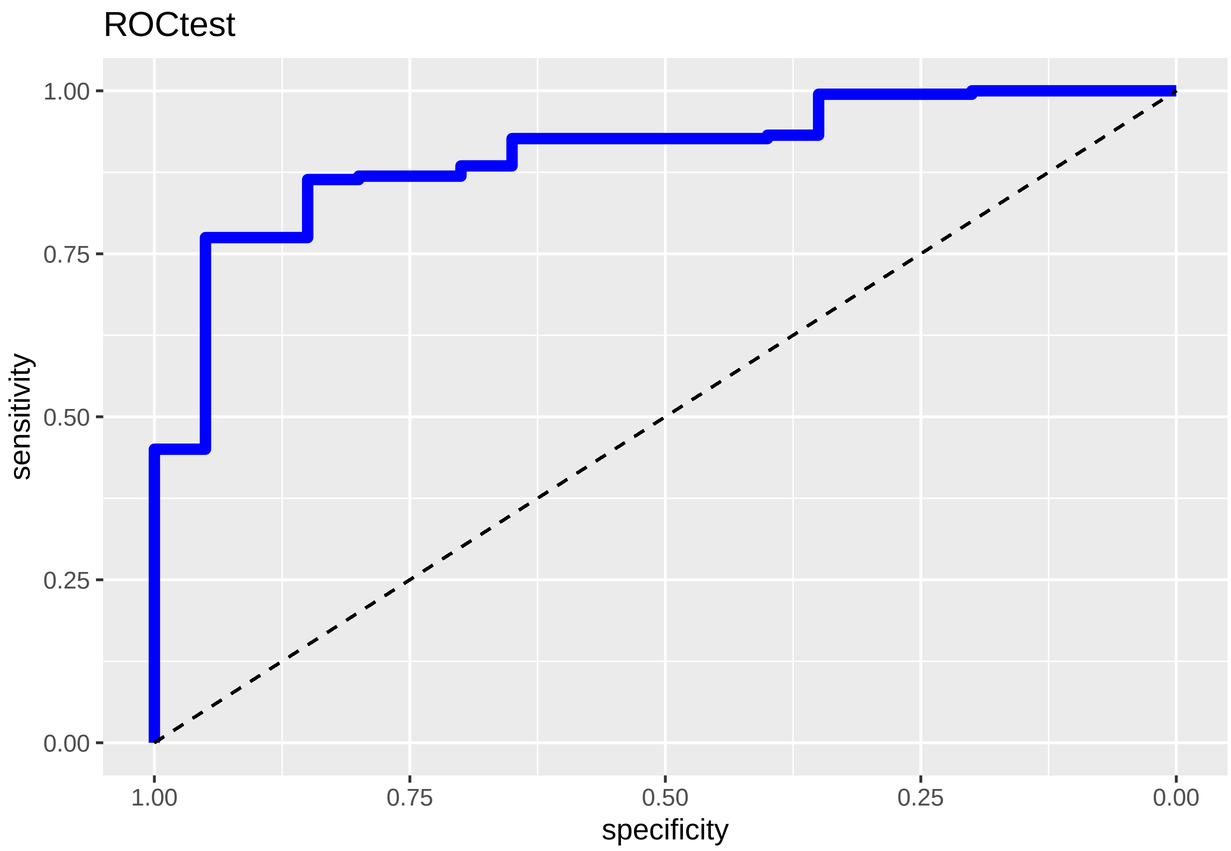
```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[2]] <- ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[2]]
```

```
##       y_pred
## y_true   0   1
##      0   7  13
##      1   3 188
```

```
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, 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 + marginVictoryTeam + nrtgTeamMisc +
                    winsTeam + pctEFGTeamMisc + pctFG2PerGameTeam +
                    pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                    ortgTeamMisc + drtgTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameOpponent +
                    blkPerGameOpponent + ageMeanMisc +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
```

```
ConfusMatknn[[2]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[2]]
```

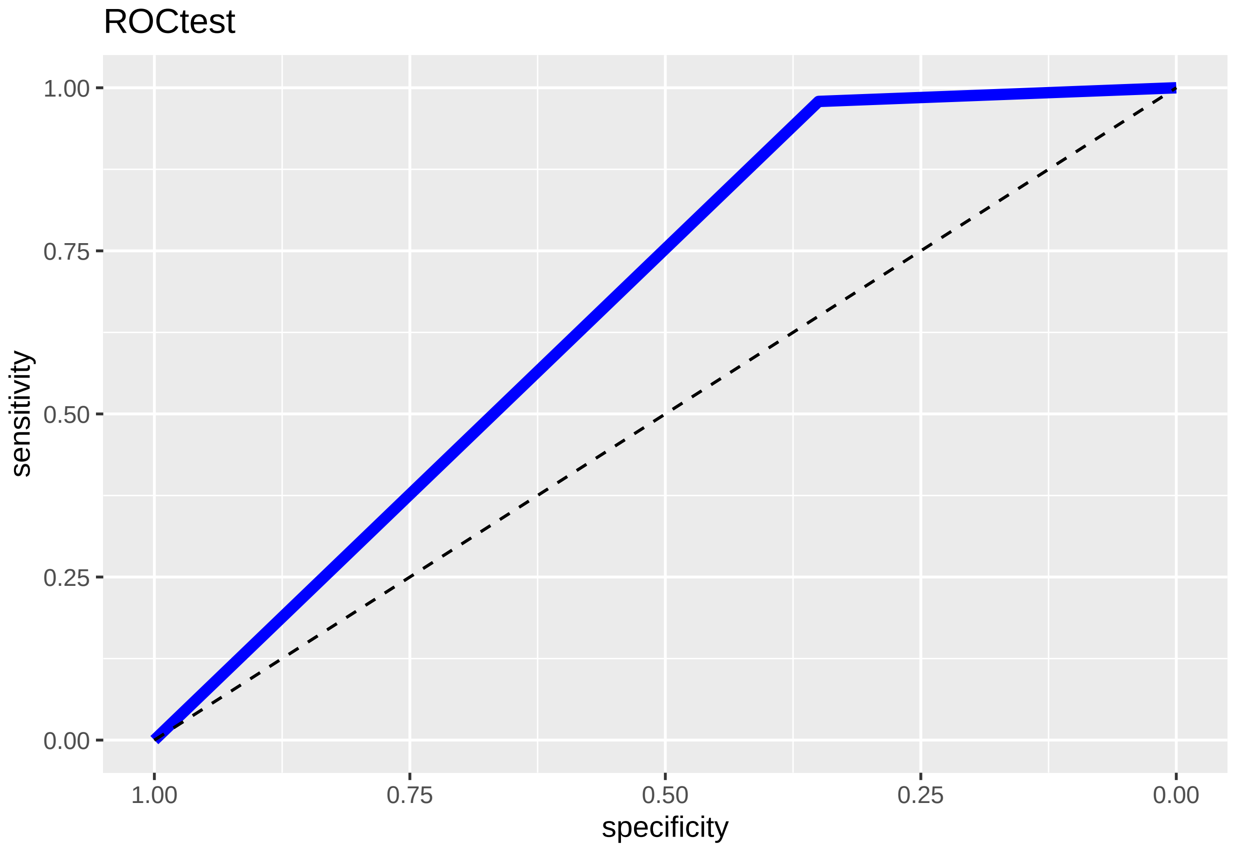
```
##       y_pred
## y_true   0   1
##      0   7  13
##      1   4 187
```

```
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()
```



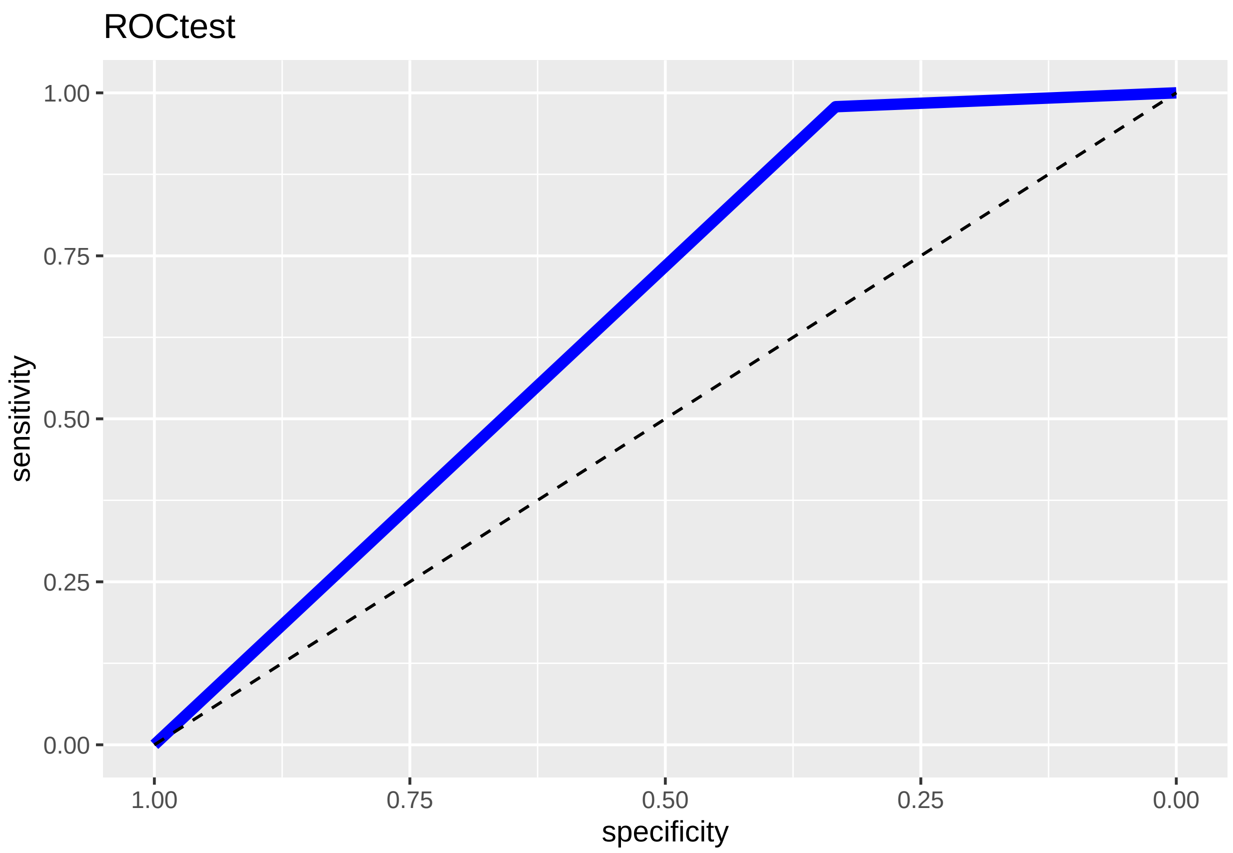
```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

##3rd fold = validation set
ytrain <- ceiling((rbind(cbind(PerGameFold1[,3]),cbind(PerGameFold2[,3]),cbind(PerGameFold4[,3]))-1)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold2[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold3[,3]-1)/5)
xtest <- cbind(PerGameFold3[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~Ranking + tovPerGameOpponent + blkPerGameOpponent +
                   fg3mPerGameTeam + ptsPerGameOpponent +
                   ftmPerGameOpponent + pctTOVOpponentMisc,
                 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   8  16
##      1   4 184
```

```
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 + marginVictoryTeam + nrtgTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc +
                   pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                   pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
                   pctFG2PerGameOpponent + ageMeanMisc + blkPerGameOpponent +
                   ptsPerGameOpponent + drbPerGameTeam + astPerGameOpponent +
                   drbPerGameOpponent + ptsPerGameTeam, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc + pctTrueShootingeTeamMisc +
##     pctFGPerGameTeam + pctFGPerGameOpponent + ortgTeamMisc +
##     pctEFGTeamOppMisc + pctFG2PerGameOpponent + ageMeanMisc +
##     blkPerGameOpponent + ptsPerGameOpponent + drbPerGameTeam +
##     astPerGameOpponent + drbPerGameOpponent + ptsPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##          0          1
## 0.09936909 0.90063091
```

```
##
## Group means:
##       Ranking   winsTeam marginVictoryTeam nrtgTeamMisc pctEFGTeamMisc
## 0 -1.29651460  1.2757859        1.29923495   1.30236453     0.97994647
## 1  0.09729741 -0.0975023       -0.08864642  -0.08950992    -0.08241628
##   pctFG2PerGameTeam drtgTeamMisc pctTrueShootingeTeamMisc pctFGPerGameTeam
## 0        1.05105006  -1.07392226               0.93304544         1.047700
## 1       -0.06876134   0.07781461              -0.07271973        -0.078258
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctFG2PerGameOpponent
## 0          -0.94570552   0.97489927       -0.95309864            -0.8903406
## 1           0.05102073  -0.06400469        0.04875664             0.0405034
##   ageMeanMisc blkPerGameOpponent ptsPerGameOpponent drbPerGameTeam
## 0  0.76895813        -0.78797055        -0.86364556     0.60888823
## 1 -0.07176876         0.06625627         0.04199522    -0.07922323
##   astPerGameOpponent drbPerGameOpponent ptsPerGameTeam
## 0        -0.83906451        -0.83316892     0.58711315
## 1         0.02749945         0.02553129    -0.05912963
##
## Coefficients of linear discriminants:
##                                  LD1
## Ranking                   0.74290440
## winsTeam                  0.78009936
## marginVictoryTeam         1.22938049
## nrtgTeamMisc             -1.14336635
## pctEFGTeamMisc            0.17315350
## pctFG2PerGameTeam        -0.47799696
## drtgTeamMisc              0.62690102
## pctTrueShootingeTeamMisc  0.28673515
## pctFGPerGameTeam         -0.17229676
## pctFGPerGameOpponent     -0.26413943
## ortgTeamMisc              0.14816896
## pctEFGTeamOppMisc         0.49297534
## pctFG2PerGameOpponent    -0.33413839
## ageMeanMisc              -0.19388061
## blkPerGameOpponent        0.21934836
## ptsPerGameOpponent        0.13409000
## drbPerGameTeam            0.09944056
## astPerGameOpponent        0.04624626
## drbPerGameOpponent        0.39734198
## ptsPerGameTeam           -0.62096737
```

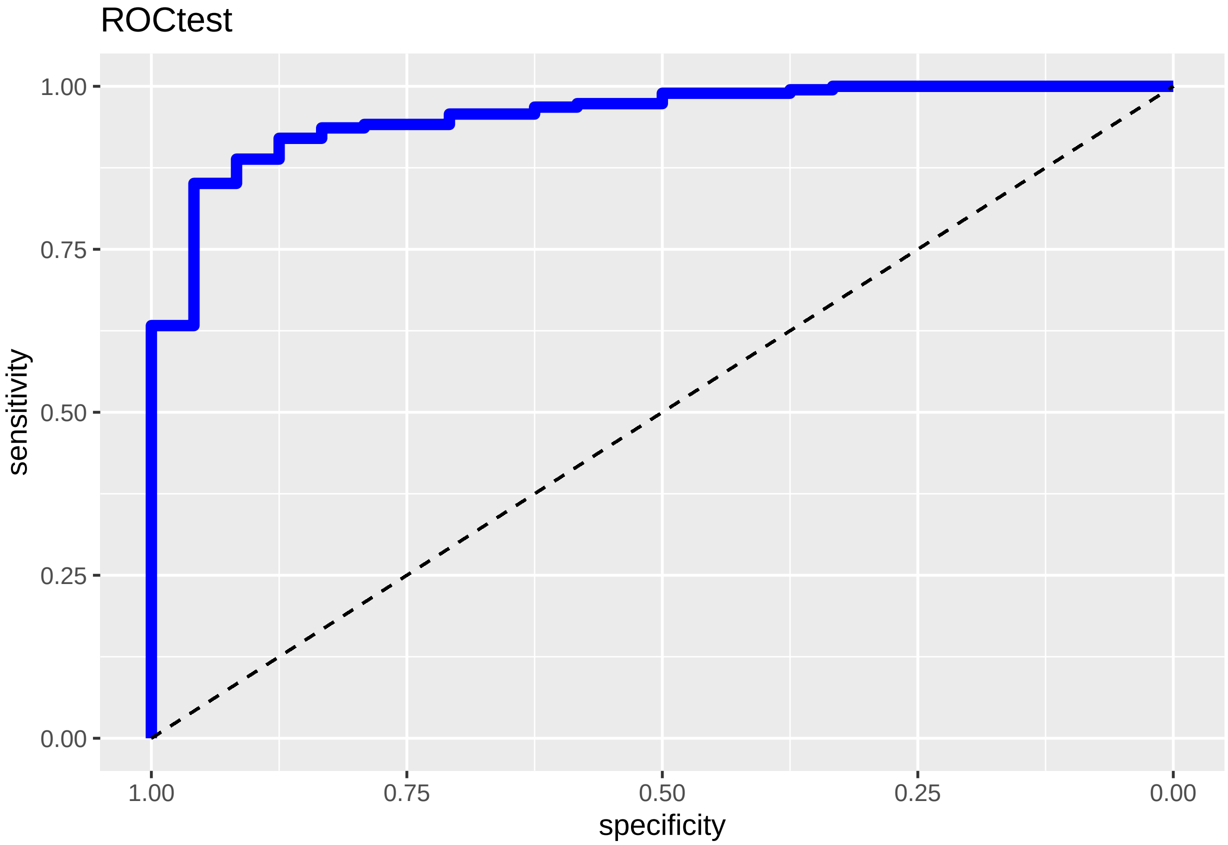
```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[3]] <- ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[3]]
```

```
##       y_pred
## y_true   0   1
##      0   5  19
##      1   0 188
```

```
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(datat
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, 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 + marginVictoryTeam + nrtgTeamMisc +
                    winsTeam + pctEFGTeamMisc + pctFG2PerGameTeam +
                    pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                    ortgTeamMisc + drtgTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameOpponent +
                    blkPerGameOpponent + ageMeanMisc +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
```

```
ConfusMatknn[[3]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[3]]
```

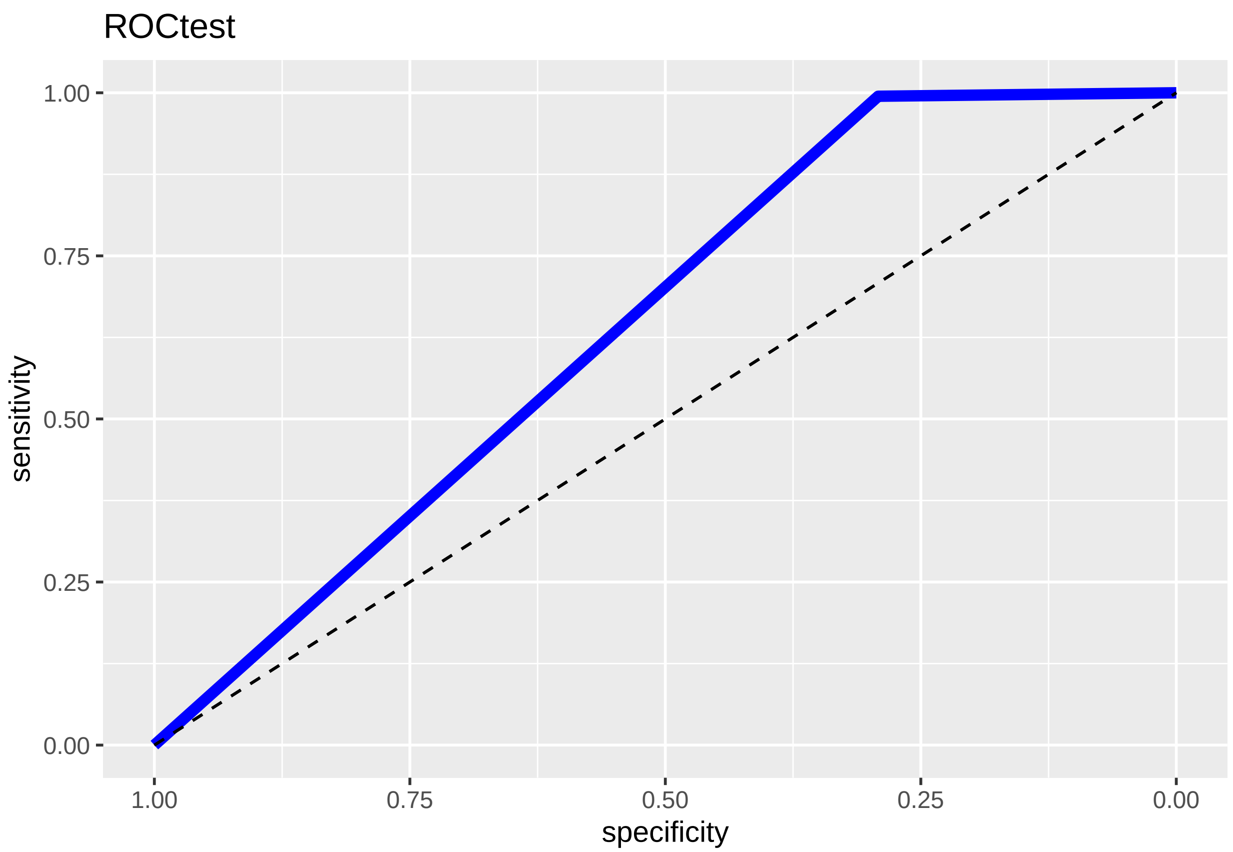
```
##       y_pred
## y_true   0   1
##      0   7  17
##      1   1 187
```

```
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$yt
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, knntest
F1knn <- F1knn + ifelse(is.nan(F1_Score(datatest$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()
```



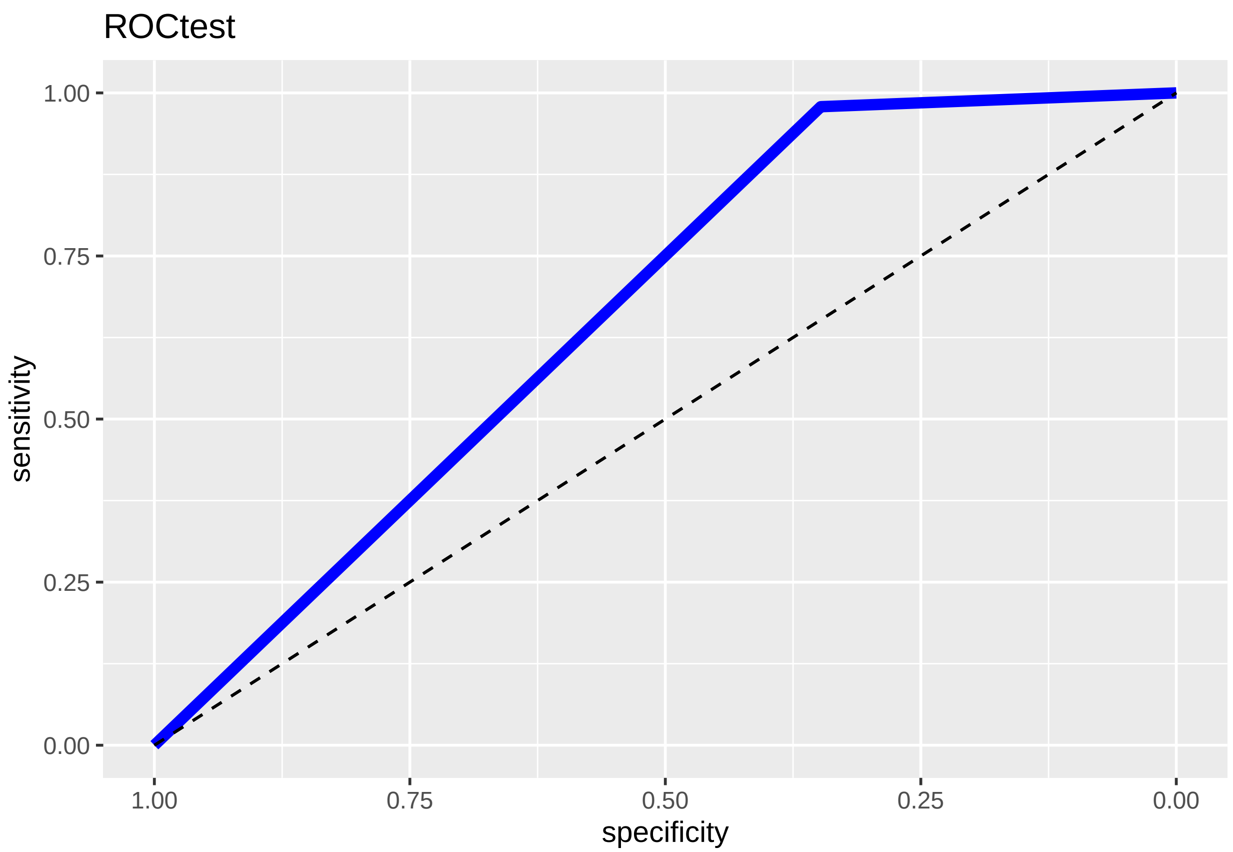
```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

##4th fold = validation set
ytrain <- ceiling((rbind(cbind(PerGameFold1[,3]),cbind(PerGameFold2[,3]),cbind(PerGameFold3[,3]))-1)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold2[,-3],PerGameFold3[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold4[,3]-1)/5)
xtest <- cbind(PerGameFold4[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~Ranking + tovPerGameOpponent + blkPerGameOpponent +
                   fg3mPerGameTeam + ptsPerGameOpponent +
                   ftmPerGameOpponent + pctTOVOpponentMisc,
                 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   8  15
##      1   4 185
```

```
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 + marginVictoryTeam + nrtgTeamMisc +
                   pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc +
                   pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                   pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
                   pctFG2PerGameOpponent + ageMeanMisc + blkPerGameOpponent +
                   ptsPerGameOpponent + drbPerGameTeam + astPerGameOpponent +
                   drbPerGameOpponent + ptsPerGameTeam, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     pctEFGTeamMisc + pctFG2PerGameTeam + drtgTeamMisc + pctTrueShootingeTeamMisc +
##     pctFGPerGameTeam + pctFGPerGameOpponent + ortgTeamMisc +
##     pctEFGTeamOppMisc + pctFG2PerGameOpponent + ageMeanMisc +
##     blkPerGameOpponent + ptsPerGameOpponent + drbPerGameTeam +
##     astPerGameOpponent + drbPerGameOpponent + ptsPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.1009464 0.8990536
```

```
##
## Group means:
##      Ranking   winsTeam marginVictoryTeam nrtgTeamMisc pctEFGTeamMisc
## 0 -1.3255434  1.3061231         1.3356546     1.340433      0.9334814
## 1  0.1162671 -0.1206345        -0.1096555    -0.109932     -0.1094098
##   pctFG2PerGameTeam drtgTeamMisc pctTrueShootingeTeamMisc pctFGPerGameTeam
## 0         1.0029525  -1.11521097                0.8337974        0.9706375
## 1        -0.1061554   0.06738608               -0.1131568       -0.1038023
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctFG2PerGameOpponent
## 0          -0.96011477    0.9714444       -0.99678214           -0.90957164
## 1           0.04672453   -0.1025307        0.04485473            0.03040653
##   ageMeanMisc blkPerGameOpponent ptsPerGameOpponent drbPerGameTeam
## 0  0.81022650        -0.86394949        -0.90690794      0.6459003
## 1 -0.08974118         0.08613642         0.02313547     -0.0960307
##   astPerGameOpponent drbPerGameOpponent ptsPerGameTeam
## 0       -0.851570560        -0.82180870      0.5700724
## 1       -0.000662564         0.01830985     -0.1053057
##
## Coefficients of linear discriminants:
##                                  LD1
## Ranking                   0.91305894
## winsTeam                  0.92986637
## marginVictoryTeam         1.53320126
## nrtgTeamMisc             -1.60988471
## pctEFGTeamMisc           -0.26966072
## pctFG2PerGameTeam        -0.48916372
## drtgTeamMisc              0.33434744
## pctTrueShootingeTeamMisc  0.75351890
## pctFGPerGameTeam          0.01876979
## pctFGPerGameOpponent     -0.38436516
## ortgTeamMisc              0.40049119
## pctEFGTeamOppMisc         0.74793752
## pctFG2PerGameOpponent    -0.45356467
## ageMeanMisc              -0.22805331
## blkPerGameOpponent        0.23136602
## ptsPerGameOpponent        0.27898764
## drbPerGameTeam            0.07116163
## astPerGameOpponent        0.03993422
## drbPerGameOpponent        0.55994027
## ptsPerGameTeam           -0.93697864
```

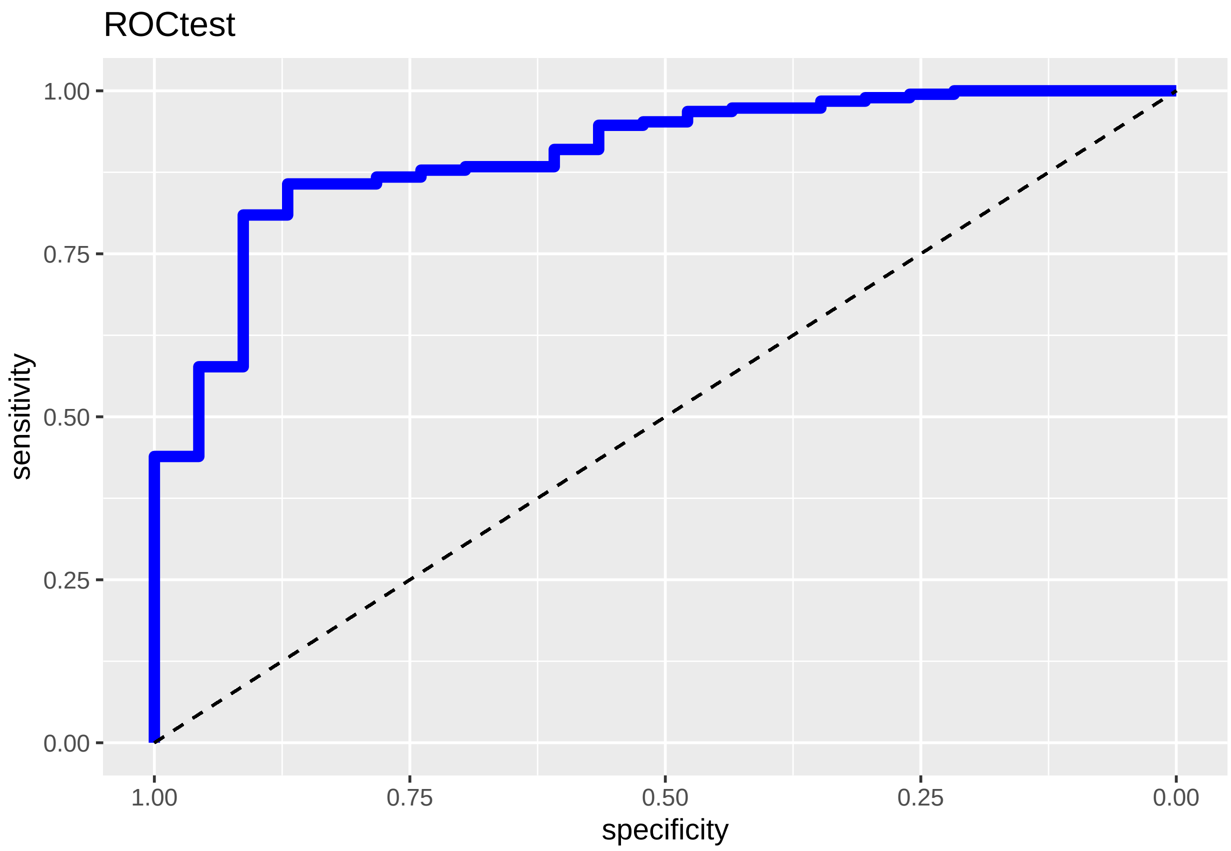
```
ldatest <- predict(model.lda, datatest)
ConfusMatlda[[4]] <- ConfusionMatrix(ldatest$class, datatest$ytest)
ConfusMatlda[[4]]
```

```
##       y_pred
## y_true   0   1
##      0   7  16
##      1   2 187
```

```
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, 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 + marginVictoryTeam + nrtgTeamMisc +
                    winsTeam + pctEFGTeamMisc + pctFG2PerGameTeam +
                    pctTrueShootingeTeamMisc + pctFGPerGameTeam +
                    ortgTeamMisc + drtgTeamMisc + pctEFGTeamOppMisc +
                    pctFGPerGameOpponent + pctFG2PerGameOpponent +
                    blkPerGameOpponent + ageMeanMisc +
                    pctFG3PerGameOpponent, data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
```

```
ConfusMatknn[[4]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[4]]
```

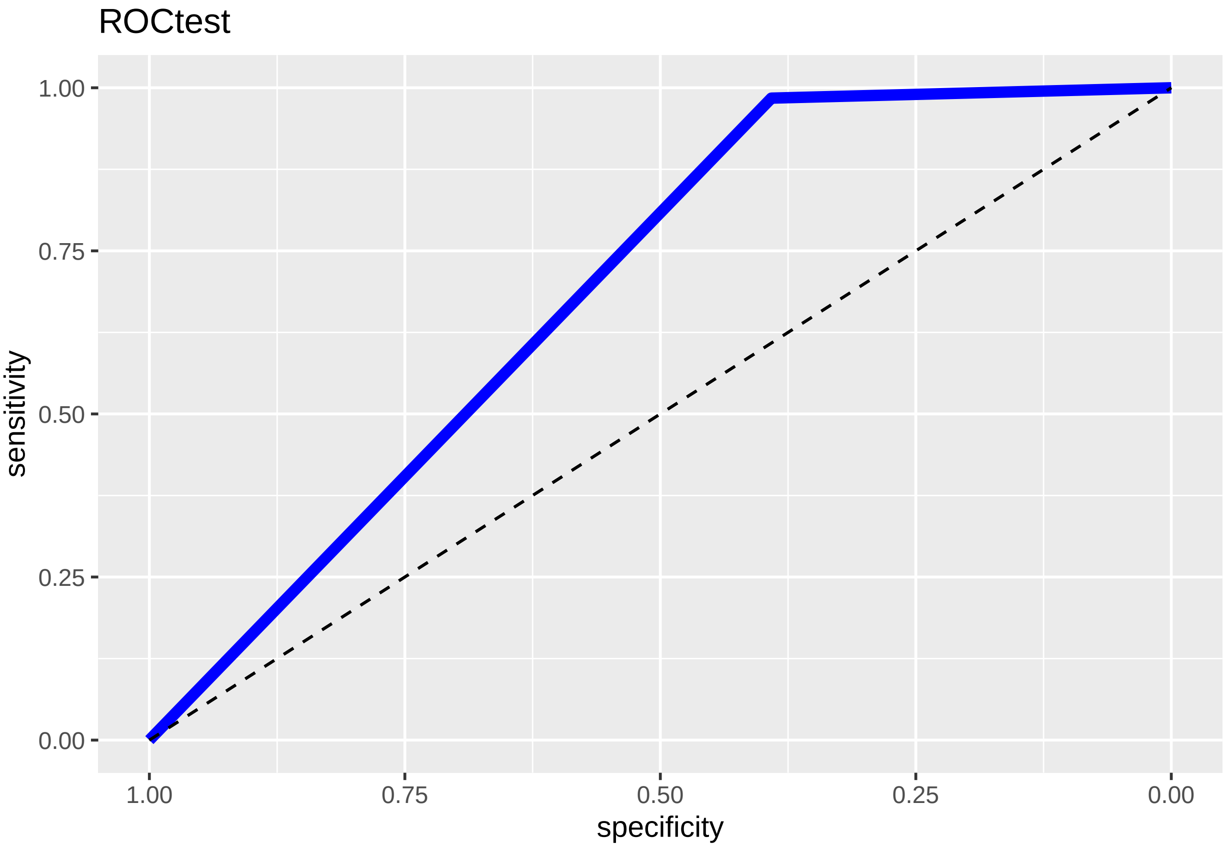
```
##       y_pred
## y_true   0   1
##      0   9  14
##      1   3 186
```

```
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] 0.08864459
```

```
Accuracyglm/4
```

```
## [1] 0.9113554
```

```
Precisionglm/4
```

```
## [1] 0.6197917
```

```
Recallglm/4
```

```
## [1] 0.3702899
```

```
F1glm/4
```

```
## [1] 0.4597718
```

```
AUCglm/4
```

```
## [1] 0.6719864
```

```
ConfusMatglm
```

```
## [[1]]
##        y_pred
## y_true   0   1
##      0   7  13
##      1   5 186
##
## [[2]]
##        y_pred
## y_true   0   1
##      0   9  11
##      1   7 184
##
## [[3]]
##        y_pred
## y_true   0   1
##      0   8  16
##      1   4 184
```

```
##
## [[4]]
##        y_pred
## y_true   0   1
##      0   8  15
##      1   4 185
```

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

```
## [1] 1.20677
```

```
Accuracylda/4
```

```
## [1] 0.9184532
```

```
Precisionlda/4
```

```
## [1] 0.7944444
```

```
Recalllda/4
```

```
## [1] 0.3031703
```

```
F1lda/4
```

```
## [1] 0.4289152
```

```
AUClda/4
```

```
## [1] 0.6463357
```

```
ConfusMatlda
```

```
## [[1]]
##        y_pred
## y_true   0   1
##      0   7  13
##      1   3 188
##
## [[2]]
##        y_pred
## y_true   0   1
##      0   7  13
##      1   3 188
##
## [[3]]
##        y_pred
## y_true   0   1
##      0   5  19
##      1   0 188
##
## [[4]]
##       y_pred
## y_true   0   1
##      0   7  16
##      1   2 187
```

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

```
## [1] 0.07800344
```

```
Accuracyknn/4
```

```
## [1] 0.9219966
```

```
Precisionknn/4
```

```
## [1] 0.7528409
```

```
Recallknn/4
```

```
## [1] 0.3707428
```

```
F1knn/4
```

```
## [1] 0.4914747
```

```
AUCknn/4
```

```
## [1] 0.6781412
```

```
ConfusMatknn
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0   9  11
##      1   3 188
##
## [[2]]
##       y_pred
## y_true   0   1
##      0   7  13
##      1   4 187
##
## [[3]]
##       y_pred
## y_true   0   1
##      0   7  17
##      1   1 187
##
## [[4]]
##       y_pred
## y_true   0   1
##      0   9  14
##      1   3 186
```

```
##2020 Season Predictions
ytrain <- ceiling((MasterPerGame$finish-1)/5)
xtrain <- MasterPerGame[,-3]
datatrain <- cbind(ytrain, xtrain)
xtest <- MasterPerGame2020

##Logistic Regression
model.glm <- glm(ytrain~Ranking + tovPerGameOpponent + blkPerGameOpponent +
                   fg3mPerGameTeam + ptsPerGameOpponent +
                   ftmPerGameOpponent + pctTOVOpponentMisc,
                 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  1  0  1  1  0  1  1  1  1  1  1  1  1  1
## 27 28 29 30
##  1  1  1  1
```

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

```
## [1] "Los Angeles Lakers"
## [1] "Milwaukee Bucks"
```

```
##Logistic Regression Model suggest that the Los Angeles Lakers and Milwaukee Bucks
##are Finalist level Teams

##Discriminant Analysis
##Linear Discriminant
```

```r
model.lda <- lda(ytrain~ winsTeam + nrtgTeamMisc + marginVictoryTeam +
                   Ranking + drtgTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam +
                   pctEFGTeamOppMisc + pctFGPerGameOpponent + pctEFGTeamMisc +
                   ortgTeamMisc + blkPerGameOpponent + pctFG2PerGameOpponent,
                 data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ winsTeam + nrtgTeamMisc + marginVictoryTeam + Ranking +
##     drtgTeamMisc + pctFG2PerGameTeam + pctFGPerGameTeam + pctEFGTeamOppMisc +
##     pctFGPerGameOpponent + pctEFGTeamMisc + ortgTeamMisc + blkPerGameOpponent +
##     pctFG2PerGameOpponent, data = datatrain)
##
## Prior probabilities of groups:
##          0          1
## 0.08510638 0.91489362
##
## Group means:
##     winsTeam nrtgTeamMisc marginVictoryTeam    Ranking drtgTeamMisc
## 0  1.3142426    1.3318791         1.3321223 -1.3696229  -1.04907303
## 1 -0.1222551   -0.1238957        -0.1239184  0.1274068   0.09758819
##   pctFG2PerGameTeam pctFGPerGameTeam pctEFGTeamOppMisc pctFGPerGameOpponent
## 0         1.1164791       1.07343808       -1.00042840          -0.98669793
## 1        -0.1038585      -0.09985471        0.09306311           0.09178585
##   pctEFGTeamMisc ortgTeamMisc blkPerGameOpponent pctFG2PerGameOpponent
## 0       1.107777   1.02378928        -0.86509663           -0.91634195
## 1      -0.103049  -0.09523621         0.08047411            0.08524111
##
## Coefficients of linear discriminants:
##                               LD1
## winsTeam               0.95695815
## nrtgTeamMisc          -0.62138748
## marginVictoryTeam     -0.10247800
## Ranking                1.04436172
## drtgTeamMisc           0.02172589
## pctFG2PerGameTeam     -0.44564019
## pctFGPerGameTeam      -0.15377145
## pctEFGTeamOppMisc      0.47933944
## pctFGPerGameOpponent  -0.19116466
## pctEFGTeamMisc         0.17450639
## ortgTeamMisc           0.18269413
## blkPerGameOpponent     0.15187168
## pctFG2PerGameOpponent -0.24339867
```

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

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 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 Lakers"
## [1] "Milwaukee Bucks"
```

```
##Discriminant Analysis Model gives the same conclusions as Logistic Regression Model

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

```
##  [1] 1 1 1 1 1 1 0 1 1 1 1 1 1 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] "Dallas Mavericks"
## [1] "Los Angeles Lakers"
```

```
##K Nearest Neighbours suggests that the Los Angeles Lakers
##are Finalist-level teams
```