# NBAAnalysisPlayoffsClassificationModel.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
##Playoffs Only Feature Selection
ytrain <- ceiling((MasterPerGame$finish-4)/5)
xtrain <- MasterPerGame[,-3]
datatrain <- cbind(ytrain, xtrain)

##Generalized Linear Model Feature Selection
set.seed(2)
cntrl <- rfeControl(functions = lrFuncs, method = "cv", number = 4, repeats = 10)
model.glm <- rfe(datatrain[,(2:63)], as.factor(datatrain[,1]), rfeControl = cntrl, sizes = c(5:25), met
```

```
## Warning in rfe.default(datatrain[, (2:63)], as.factor(datatrain[, 1]),
## rfeControl = cntrl, : Metric 'ROC' is not created by the summary function;
## 'Accuracy' will be used instead
```

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

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

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

```
model.glm
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (4 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          5   0.9126 0.8249    0.01932 0.03879
##          6   0.9114 0.8225    0.02176 0.04371
##          7   0.9066 0.8131    0.01925 0.03858
##          8   0.9090 0.8179    0.02034 0.04076
##          9   0.9078 0.8155    0.02245 0.04503
##         10   0.9114 0.8226    0.02370 0.04748
##         11   0.9114 0.8226    0.02106 0.04217
##         12   0.9114 0.8226    0.02037 0.04083
##         13   0.9149 0.8297    0.01377 0.02763
##         14   0.9161 0.8320    0.01032 0.02066        *
##         15   0.9137 0.8274    0.01634 0.03268
##         16   0.9066 0.8132    0.01542 0.03090
##         17   0.9019 0.8036    0.01293 0.02616
##         18   0.9019 0.8035    0.01400 0.02820
##         19   0.9055 0.8108    0.01266 0.02540
##         20   0.9066 0.8131    0.01341 0.02696
##         21   0.9031 0.8060    0.01962 0.03925
##         22   0.9031 0.8059    0.01452 0.02917
##         23   0.9031 0.8060    0.01593 0.03194
##         24   0.9066 0.8131    0.01542 0.03093
##         25   0.9055 0.8106    0.01829 0.03672
##         62   0.8913 0.7822    0.03572 0.07171
##
## The top 5 variables (out of 14):
##    Ranking, blkPerGameTeam, pctEFGTeamMisc, drtgTeamMisc, pctFGPerGameTeam
```

```
model.glm$optVariables
```

```
##  [1] "Ranking"                 "blkPerGameTeam"
##  [3] "pctEFGTeamMisc"          "drtgTeamMisc"
##  [5] "pctFGPerGameTeam"        "nrtgTeamMisc"
##  [7] "ortgTeamMisc"            "marginVictoryTeam"
```

```
##  [9] "winsTeam"                 "pctEFGTeamOppMisc"
## [11] "pctTrueShootingeTeamMisc" "fg2aPerGameOpponent"
## [13] "fg2mPerGameOpponent"      "stlPerGameTeam"
```

```
##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   0.9090 0.8178   0.011624 0.023255
##          6   0.9055 0.8106   0.013779 0.027615
##          7   0.9043 0.8083   0.014533 0.029082
##          8   0.9055 0.8107   0.013779 0.027548
##          9   0.9066 0.8131   0.014003 0.028050
##         10   0.9125 0.8250   0.009645 0.019200
##         11   0.9173 0.8345   0.019275 0.038486
##         12   0.9161 0.8320   0.015404 0.030795
##         13   0.9161 0.8321   0.013923 0.027798
##         14   0.9220 0.8439   0.012330 0.024610        *
##         15   0.9185 0.8368   0.012813 0.025591
##         16   0.9173 0.8344   0.013970 0.027919
##         17   0.9149 0.8296   0.007559 0.015065
##         18   0.9161 0.8320   0.007879 0.015719
##         19   0.9149 0.8297   0.005227 0.010361
##         20   0.9125 0.8250   0.002491 0.004919
##         21   0.9125 0.8250   0.002491 0.004888
##         22   0.9114 0.8226   0.004307 0.008525
##         23   0.9125 0.8250   0.008842 0.017594
##         24   0.9137 0.8274   0.007880 0.015672
##         25   0.9102 0.8202   0.010025 0.020005
##         62   0.9090 0.8179   0.016833 0.033565
##
## The top 5 variables (out of 14):
##    Ranking, winsTeam, marginVictoryTeam, nrtgTeamMisc, drtgTeamMisc
```

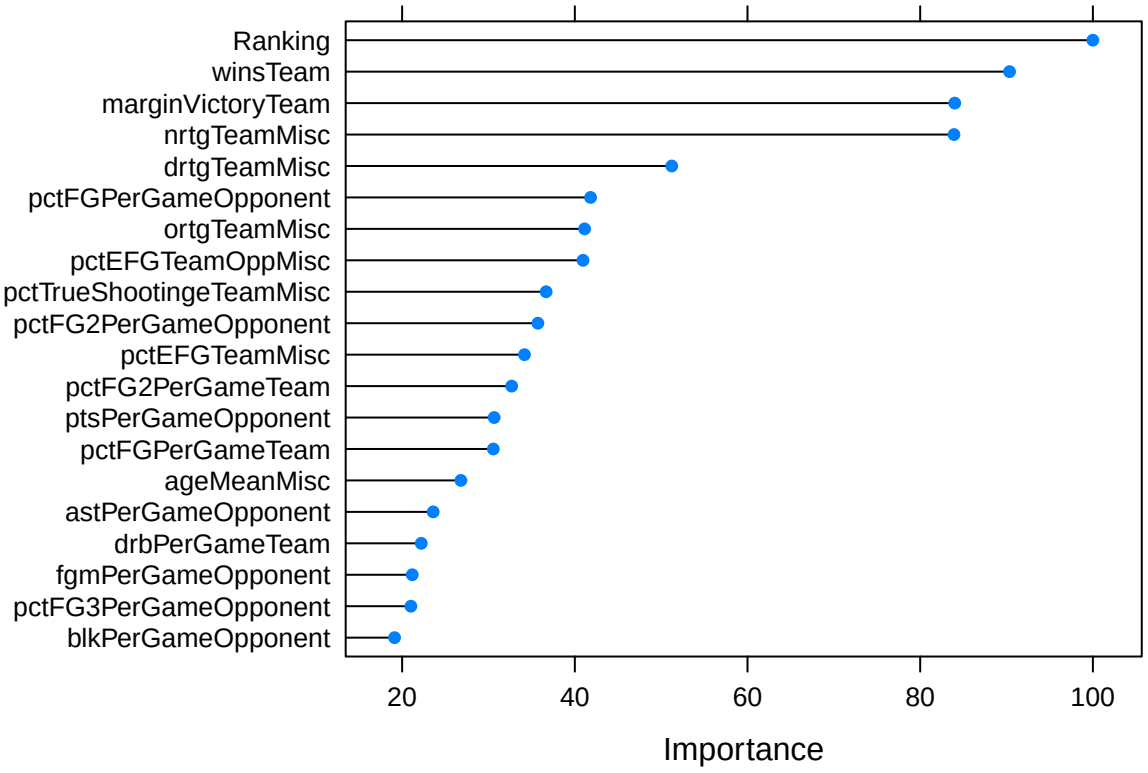
```
model.lda$optVariables
```

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

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

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 62)
##
##                           Overall
## Ranking                    100.00
## winsTeam                    90.35
## marginVictoryTeam           84.00
## nrtgTeamMisc                83.91
## drtgTeamMisc                51.23
## pctFGPerGameOpponent        41.82
## ortgTeamMisc                41.13
## pctEFGTeamOppMisc           40.95
## pctTrueShootingeTeamMisc    36.68
## pctFG2PerGameOpponent       35.73
## pctEFGTeamMisc              34.17
## pctFG2PerGameTeam           32.69
## ptsPerGameOpponent          30.66
## pctFGPerGameTeam            30.56
## ageMeanMisc                 26.81
## astPerGameOpponent          23.60
## drbPerGameTeam              22.21
## fgmPerGameOpponent          21.17
## pctFG3PerGameOpponent       21.01
## blkPerGameOpponent          19.12
```

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




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

##Playoffs Only 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]))-4)/5)
```

```r
xtrain <- rbind(PerGameFold2[,-3],PerGameFold3[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold1[,3]-4)/5)
xtest <- cbind(PerGameFold1[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
model.glm <- glm(ytrain~ Ranking + blkPerGameTeam + pctEFGTeamMisc +
                   nrtgTeamMisc + drtgTeamMisc + pctFGPerGameTeam +
                   ortgTeamMisc + pctTrueShootingeTeamMisc +
                   marginVictoryTeam + pctEFGTeamOppMisc + winsTeam +
                   stlPerGameTeam + fg2aPerGameOpponent + fg2mPerGameOpponent,
                 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]]
```

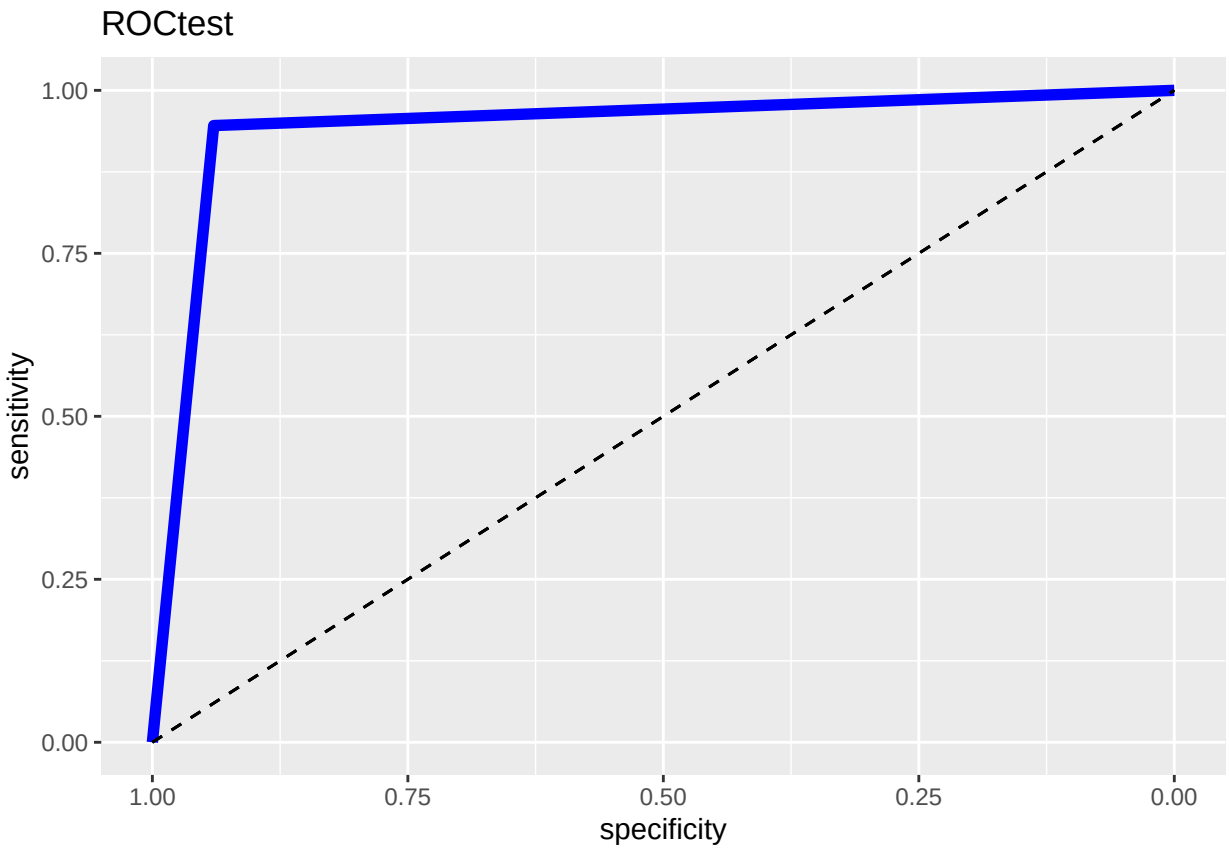
```
##        y_pred
## y_true   0   1
##      0  94   6
##      1   6 105
```

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

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     drtgTeamMisc + pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
##     pctTrueShootingeTeamMisc + pctFG2PerGameOpponent + pctEFGTeamMisc +
##     pctFG2PerGameTeam + ptsPerGameOpponent + pctFGPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.5055118 0.4944882
##
## Group means:
##      Ranking   winsTeam marginVictoryTeam nrtgTeamMisc drtgTeamMisc
## 0 -0.8726838  0.8074641         0.7731998    0.7744736   -0.6586363
```

```
## 1  0.7994116 -0.7562388        -0.7234446   -0.7242542    0.5952694
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctTrueShootingeTeamMisc
## 0           -0.6170025    0.5447117        -0.6142525                0.4784154
## 1            0.5266197   -0.5279122         0.5162325               -0.4761988
##   pctFG2PerGameOpponent pctEFGTeamMisc pctFG2PerGameTeam ptsPerGameOpponent
## 0            -0.6016130      0.4829140         0.4947230         -0.5161449
## 1             0.4826653     -0.4612211        -0.4626908          0.4311504
##   pctFGPerGameTeam
## 0        0.4685001
## 1       -0.4465077
##
## Coefficients of linear discriminants:
##                                    LD1
## Ranking                    2.063491770
## winsTeam                   0.168677712
## marginVictoryTeam         -2.099066946
## nrtgTeamMisc               1.543978272
## drtgTeamMisc              -0.182884354
## pctFGPerGameOpponent      -0.015722786
## ortgTeamMisc               0.410656808
## pctEFGTeamOppMisc         -0.007893177
## pctTrueShootingeTeamMisc -0.297079188
## pctFG2PerGameOpponent     -0.007910125
## pctEFGTeamMisc             0.678610301
## pctFG2PerGameTeam         -0.205939607
## ptsPerGameOpponent        -0.189666168
## pctFGPerGameTeam          -0.092933768
```

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

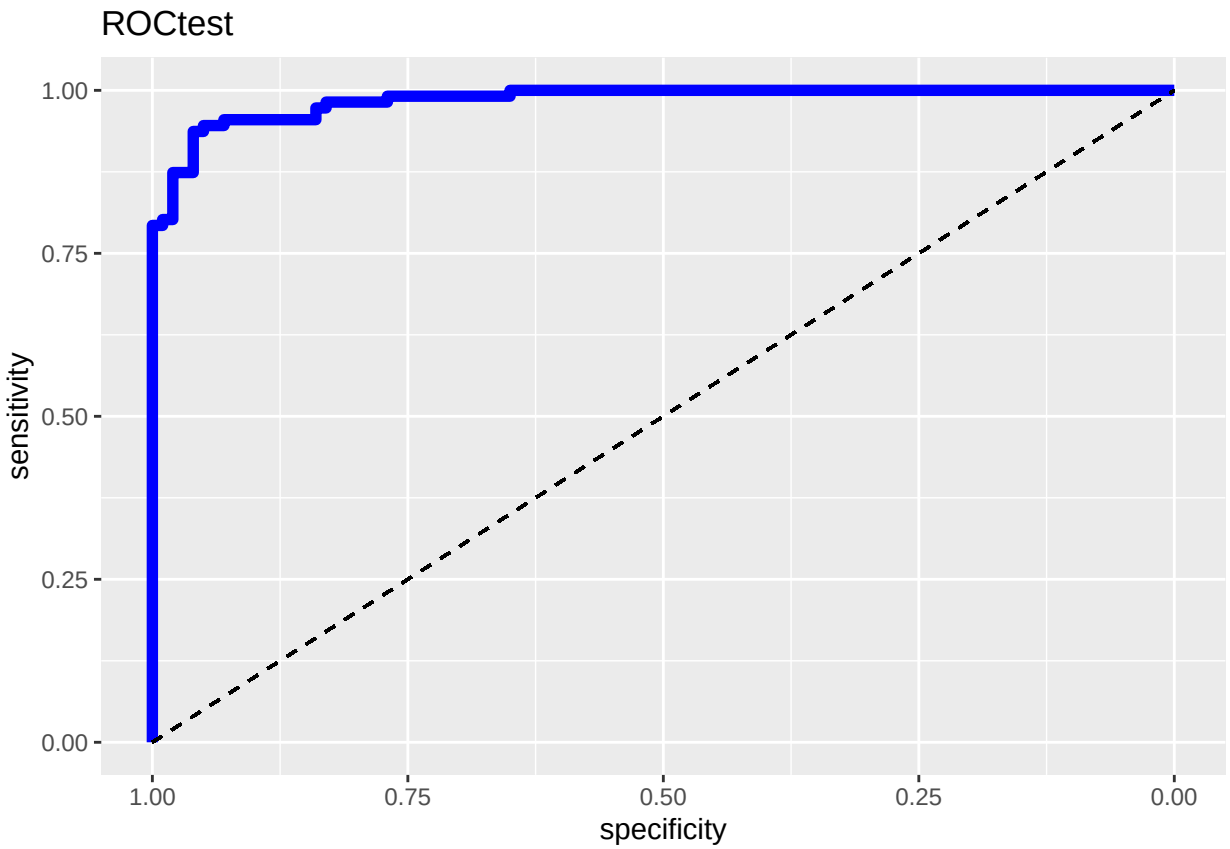
```
##       y_pred
## y_true   0   1
##      0  96   4
##      1   7 104
```

```
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 + marginVictoryTeam +
                    nrtgTeamMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctEFGTeamOppMisc + pctTrueShootingeTeamMisc +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + ageMeanMisc +
                    astPerGameOpponent + drbPerGameTeam +
                    fgmPerGameOpponent + 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 91  9
##      1 17 94
```

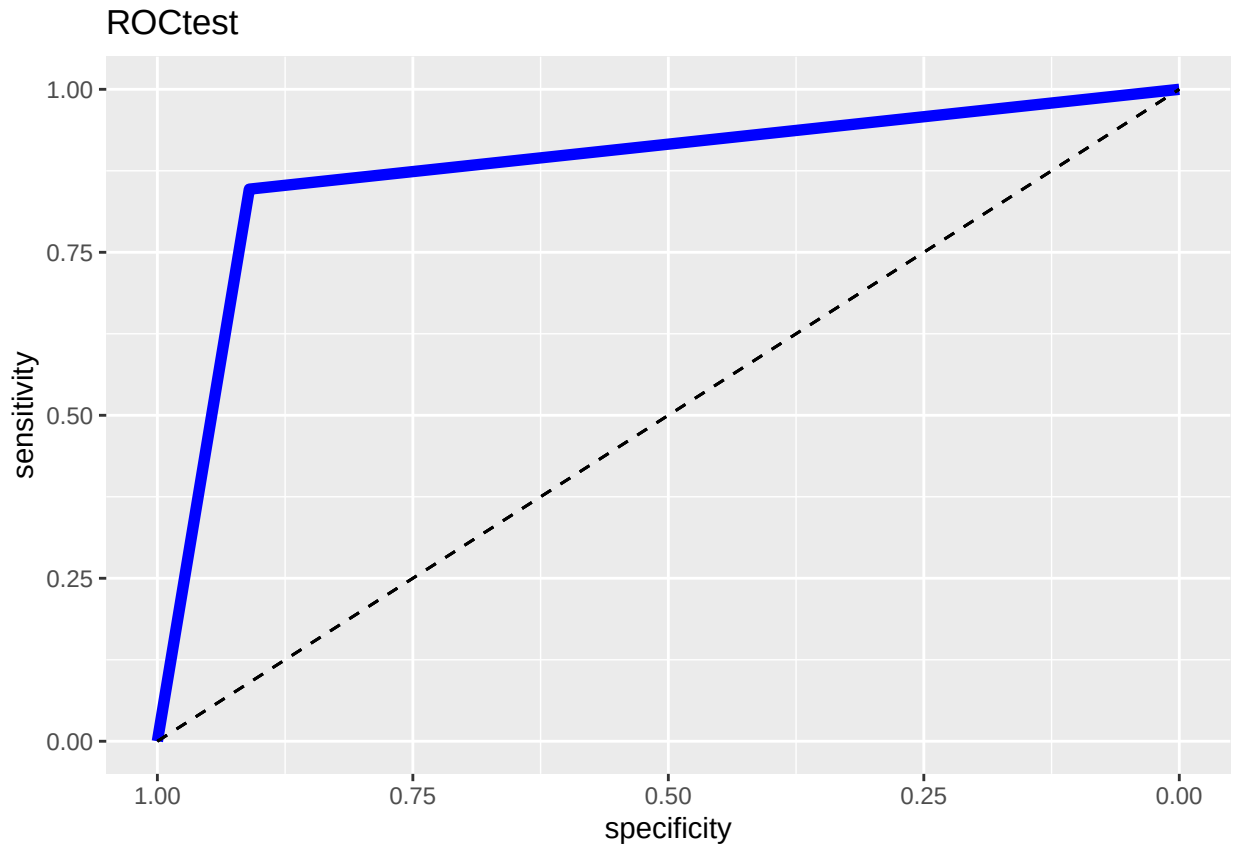
```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, 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]))-4)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold3[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold2[,3]-4)/5)
xtest <- cbind(PerGameFold2[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
```

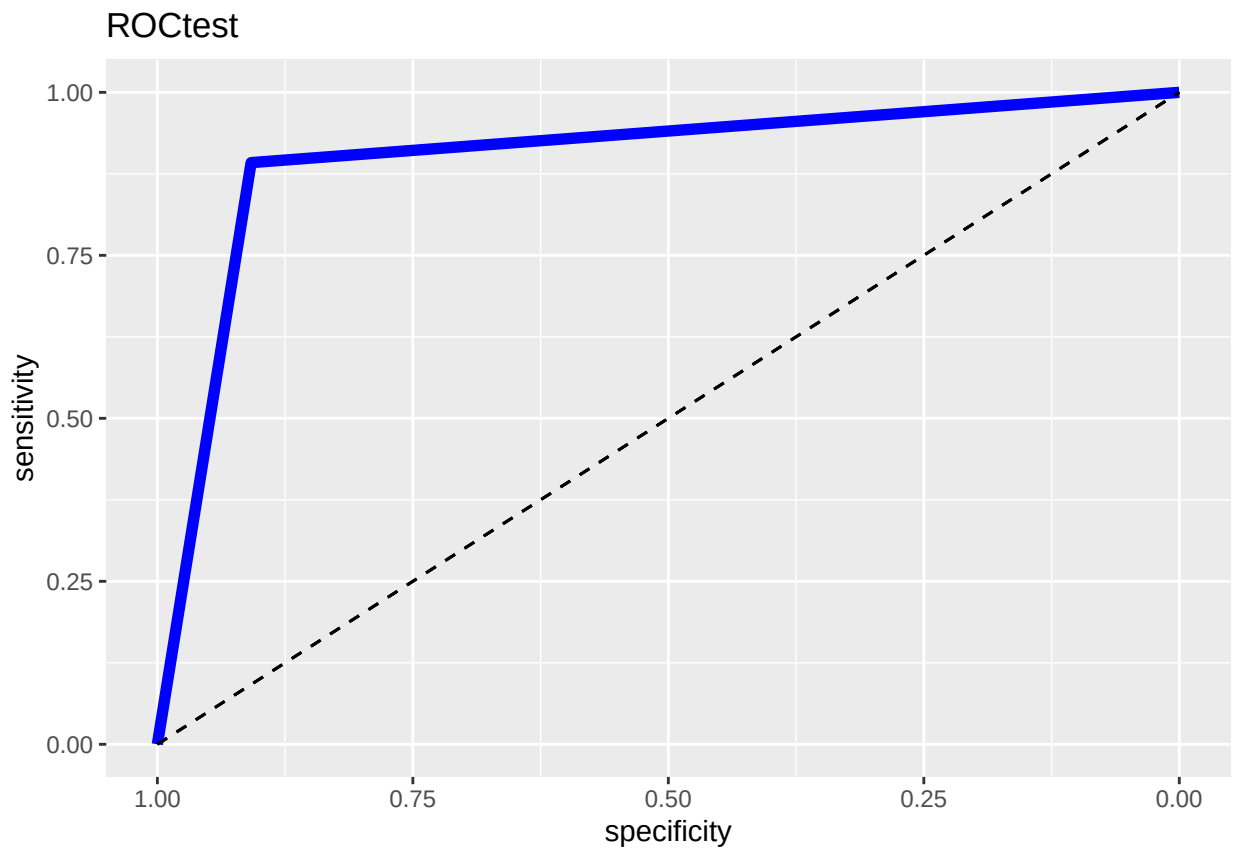
```
model.glm <- glm(ytrain~ Ranking + blkPerGameTeam + pctEFGTeamMisc +
                   nrtgTeamMisc + drtgTeamMisc + pctFGPerGameTeam +
                   ortgTeamMisc + pctTrueShootingeTeamMisc +
                   marginVictoryTeam + pctEFGTeamOppMisc + winsTeam +
                   stlPerGameTeam + fg2aPerGameOpponent + fg2mPerGameOpponent,
                 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 99 10
##      1 11 91
```

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

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     drtgTeamMisc + pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
##     pctTrueShootingeTeamMisc + pctFG2PerGameOpponent + pctEFGTeamMisc +
##     pctFG2PerGameTeam + ptsPerGameOpponent + pctFGPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.4913386 0.5086614
##
## Group means:
##      Ranking   winsTeam marginVictoryTeam nrtgTeamMisc drtgTeamMisc
## 0 -0.8701317  0.8110858         0.7858878    0.7867727   -0.6495585
```

```
## 1  0.7911940 -0.7468053        -0.7114531   -0.7114535    0.5556678
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctTrueShootingeTeamMisc
## 0           -0.6036800    0.5692426        -0.6093024                0.5004283
## 1            0.5147636   -0.5334757         0.5125389               -0.5060447
##   pctFG2PerGameOpponent pctEFGTeamMisc pctFG2PerGameTeam ptsPerGameOpponent
## 0            -0.5841209      0.4916338         0.4774627         -0.5065923
## 1             0.4694125     -0.4888996        -0.4720688          0.4298918
##   pctFGPerGameTeam
## 0        0.4583525
## 1       -0.4637284
##
## Coefficients of linear discriminants:
##                                    LD1
## Ranking                    1.958687914
## winsTeam                   0.029380581
## marginVictoryTeam         -0.979082785
## nrtgTeamMisc               0.461758288
## drtgTeamMisc              -0.240961288
## pctFGPerGameOpponent      -0.056282086
## ortgTeamMisc               0.372140337
## pctEFGTeamOppMisc          0.087919738
## pctTrueShootingeTeamMisc -0.414174627
## pctFG2PerGameOpponent     -0.002821959
## pctEFGTeamMisc             0.696580907
## pctFG2PerGameTeam         -0.051876080
## ptsPerGameOpponent        -0.188373612
## pctFGPerGameTeam          -0.097979627
```

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

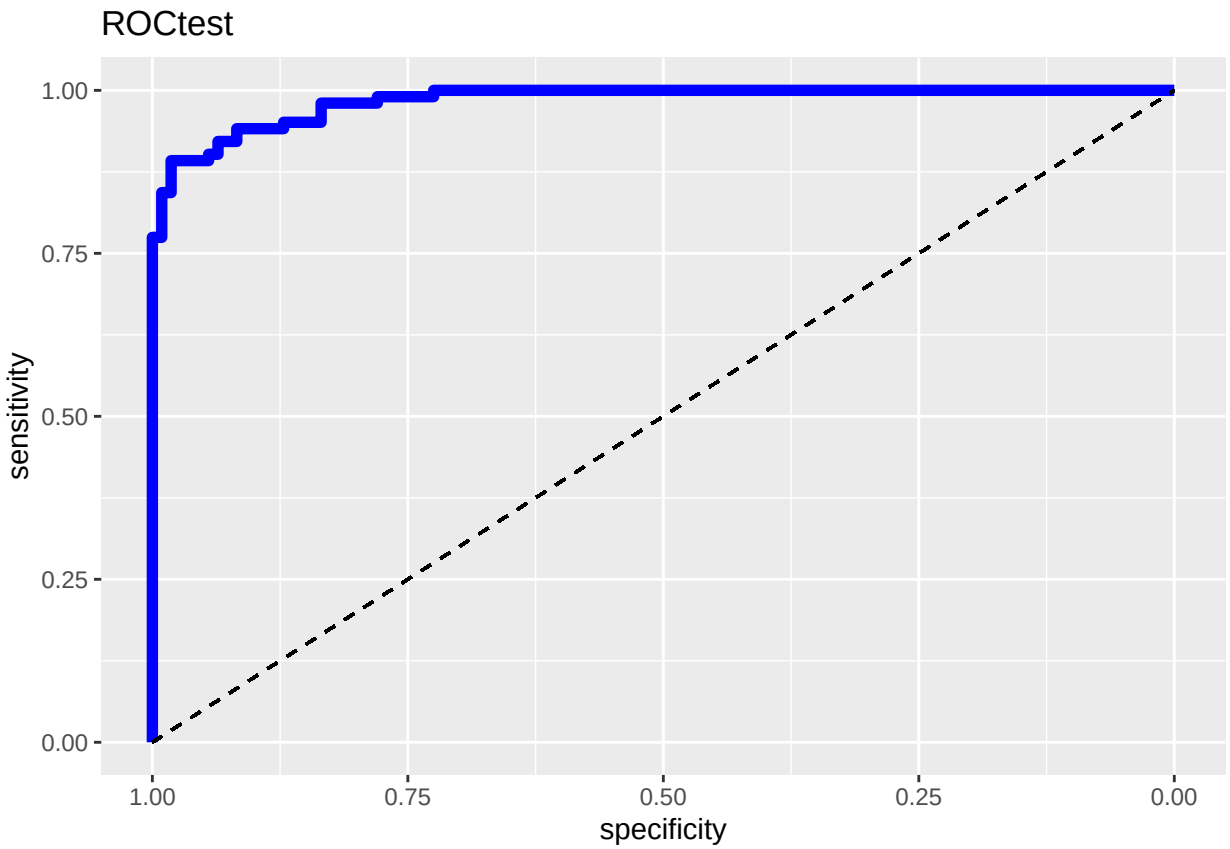
```
##       y_pred
## y_true   0   1
##      0 102   7
##      1   8  94
```

```
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 + marginVictoryTeam +
                    nrtgTeamMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctEFGTeamOppMisc + pctTrueShootingeTeamMisc +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + ageMeanMisc +
                    astPerGameOpponent + drbPerGameTeam +
                    fgmPerGameOpponent + 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
##      0 100   9
##      1  13  89
```

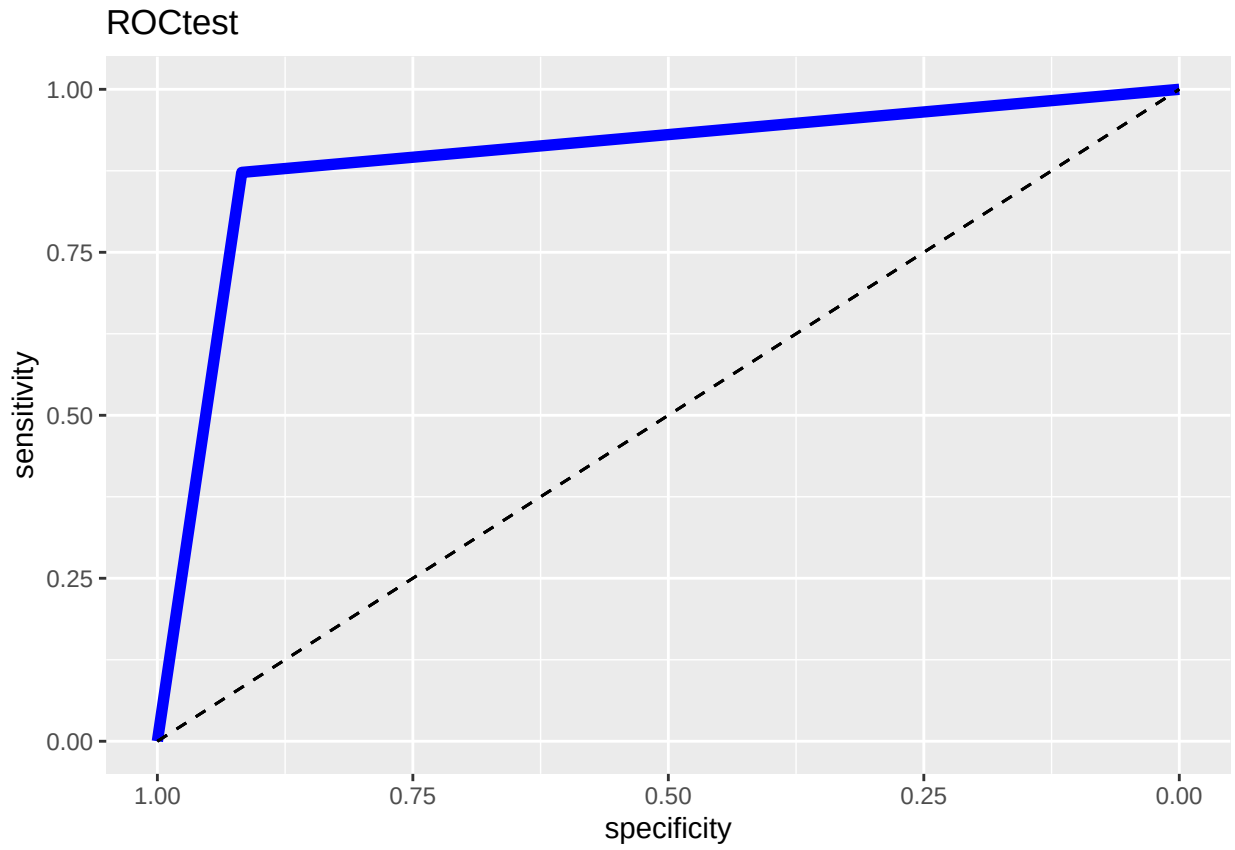
```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, 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]))-4)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold2[,-3],PerGameFold4[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold3[,3]-4)/5)
xtest <- cbind(PerGameFold3[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
```

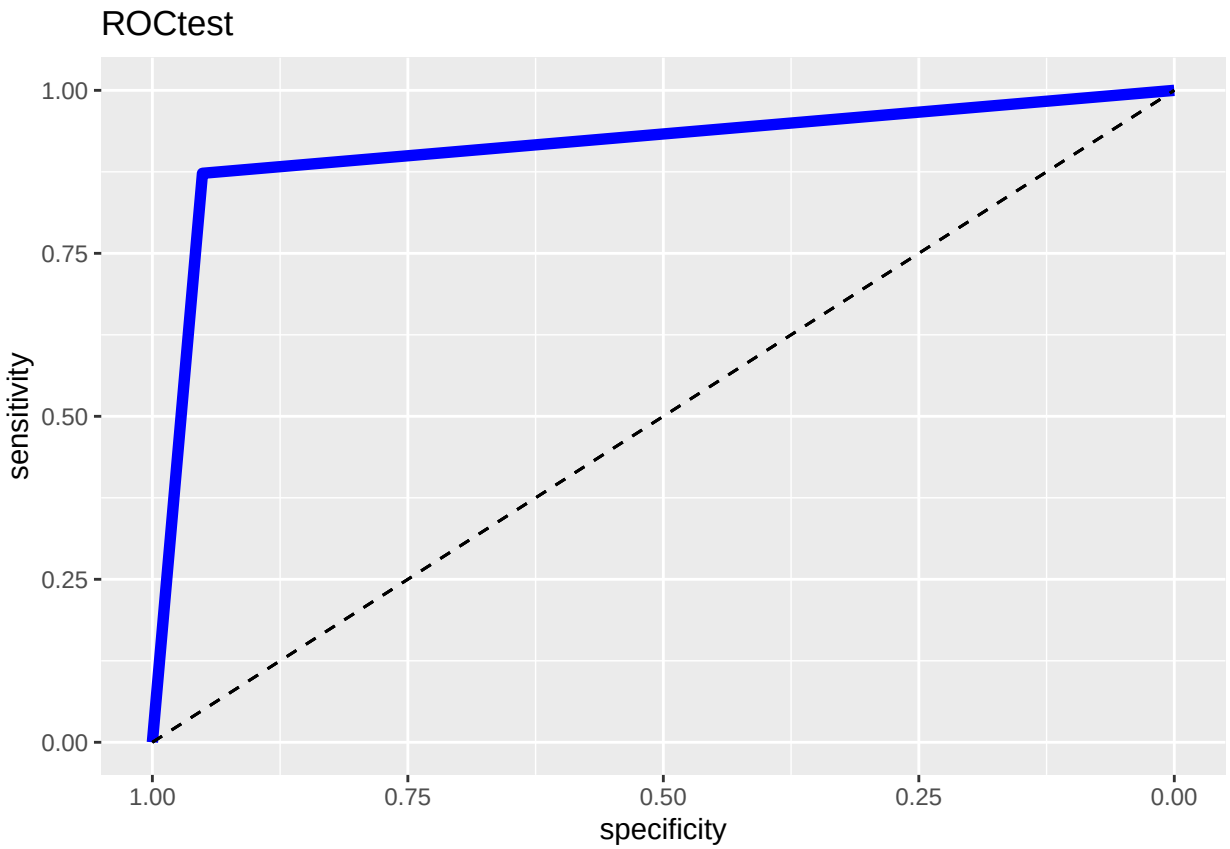
```
model.glm <- glm(ytrain~ Ranking + blkPerGameTeam + pctEFGTeamMisc +
                   nrtgTeamMisc + drtgTeamMisc + pctFGPerGameTeam +
                   ortgTeamMisc + pctTrueShootingeTeamMisc +
                   marginVictoryTeam + pctEFGTeamOppMisc + winsTeam +
                   stlPerGameTeam + fg2aPerGameOpponent + fg2mPerGameOpponent,
                 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 97  5
##      1 14 96
```

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

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     drtgTeamMisc + pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
##     pctTrueShootingeTeamMisc + pctFG2PerGameOpponent + pctEFGTeamMisc +
##     pctFG2PerGameTeam + ptsPerGameOpponent + pctFGPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.5031546 0.4968454
##
## Group means:
##      Ranking   winsTeam marginVictoryTeam nrtgTeamMisc drtgTeamMisc
## 0 -0.8761141  0.8194564         0.7970069    0.7980741   -0.6405541
```

```
## 1  0.8043072 -0.7514473        -0.7079699   -0.7099899    0.5749581
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctTrueShootingeTeamMisc
## 0           -0.5975514    0.5971910        -0.5913159                0.5360149
## 1            0.5084834   -0.5258157         0.4965860               -0.4880313
##   pctFG2PerGameOpponent pctEFGTeamMisc pctFG2PerGameTeam ptsPerGameOpponent
## 0            -0.5656540      0.5349785         0.5385292         -0.5048412
## 1             0.4681892     -0.4951785        -0.4598013          0.4146474
##   pctFGPerGameTeam
## 0        0.5183754
## 1       -0.4572762
##
## Coefficients of linear discriminants:
##                                  LD1
## Ranking                   2.09427438
## winsTeam                 -0.03780318
## marginVictoryTeam        -1.30028882
## nrtgTeamMisc              1.22998588
## drtgTeamMisc              0.03391074
## pctFGPerGameOpponent      0.13925596
## ortgTeamMisc              0.27054422
## pctEFGTeamOppMisc        -0.20944646
## pctTrueShootingeTeamMisc -0.11933971
## pctFG2PerGameOpponent    -0.01560816
## pctEFGTeamMisc            0.44022262
## pctFG2PerGameTeam        -0.16941140
## ptsPerGameOpponent       -0.16730162
## pctFGPerGameTeam         -0.11559907
```

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

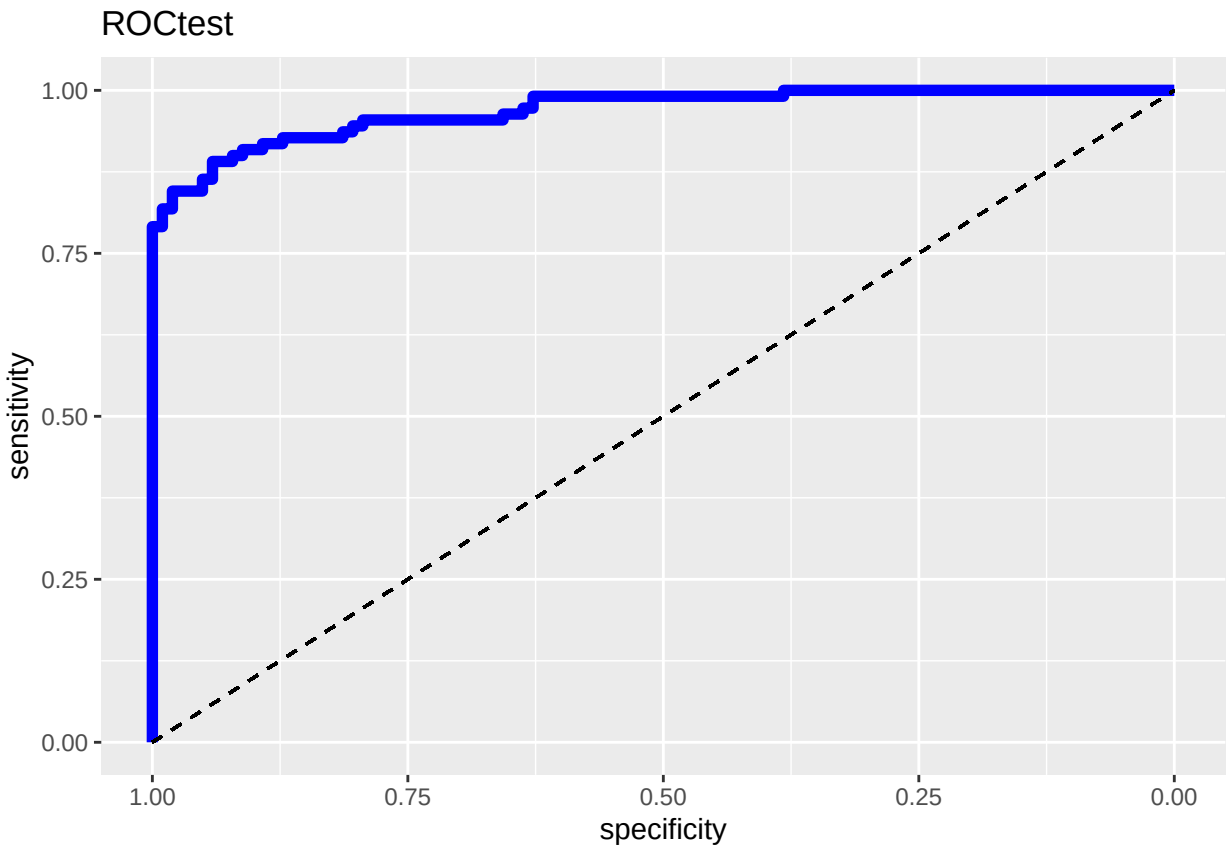
```
##       y_pred
## y_true  0  1
##      0 96  6
##      1 14 96
```

```
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 + marginVictoryTeam +
                    nrtgTeamMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctEFGTeamOppMisc + pctTrueShootingeTeamMisc +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + ageMeanMisc +
                    astPerGameOpponent + drbPerGameTeam +
                    fgmPerGameOpponent + 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 93  9
##      1 16 94
```

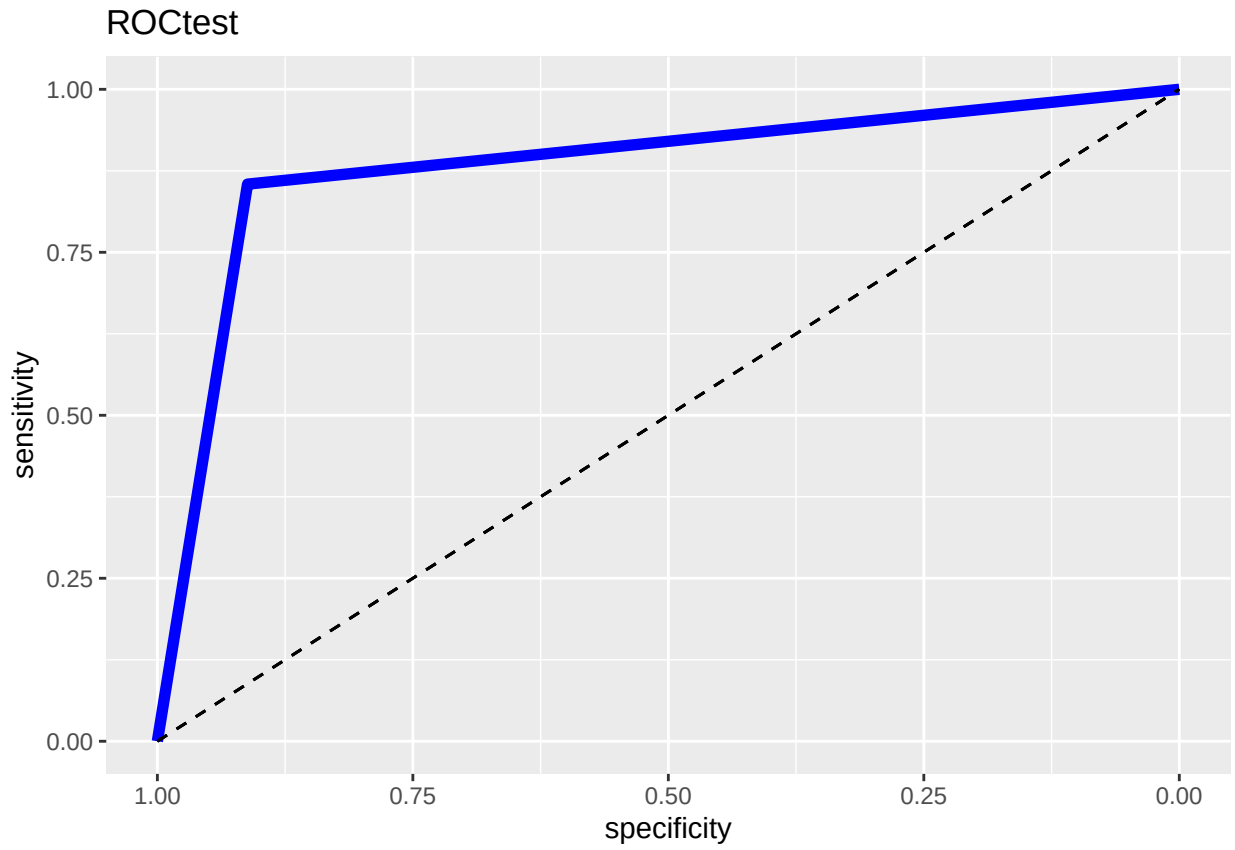
```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, 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]))-4)/5)
xtrain <- rbind(PerGameFold1[,-3],PerGameFold2[,-3],PerGameFold3[,-3])
datatrain <- cbind(ytrain, xtrain)
ytest <- ceiling((PerGameFold4[,3]-4)/5)
xtest <- cbind(PerGameFold4[,-3])
datatest <- cbind(ytest, xtest)

##Logistic Regression
```

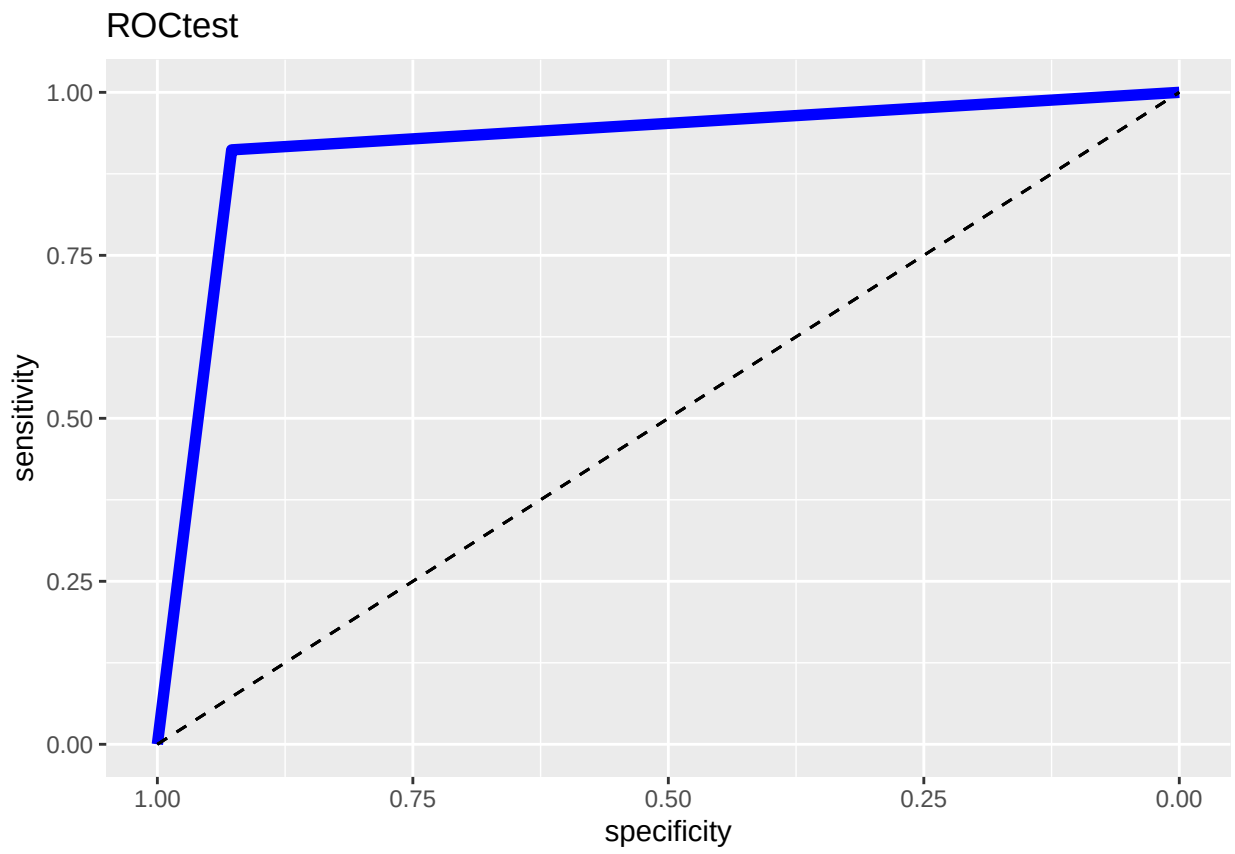
```
model.glm <- glm(ytrain~ Ranking + blkPerGameTeam + pctEFGTeamMisc +
                   nrtgTeamMisc + drtgTeamMisc + pctFGPerGameTeam +
                   ortgTeamMisc + pctTrueShootingeTeamMisc +
                   marginVictoryTeam + pctEFGTeamOppMisc + winsTeam +
                   stlPerGameTeam + fg2aPerGameOpponent + fg2mPerGameOpponent,
                 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
##      0 102   8
##      1   9  93
```

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

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     drtgTeamMisc + pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
##     pctTrueShootingeTeamMisc + pctFG2PerGameOpponent + pctEFGTeamMisc +
##     pctFG2PerGameTeam + ptsPerGameOpponent + pctFGPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.4905363 0.5094637
##
## Group means:
##      Ranking   winsTeam marginVictoryTeam nrtgTeamMisc drtgTeamMisc
## 0 -0.8774024  0.8224144         0.8096989    0.8112446   -0.6820561
```

```
## 1  0.7873362 -0.7459463        -0.7084772   -0.7095065    0.5546625
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctTrueShootingeTeamMisc
## 0           -0.6317763    0.5614492        -0.6366820                0.4802304
## 1            0.5005203   -0.5290426         0.4946788               -0.4968668
##   pctFG2PerGameOpponent pctEFGTeamMisc pctFG2PerGameTeam ptsPerGameOpponent
## 0            -0.6053068      0.4920813         0.4795001          -0.556027
## 1             0.4562525     -0.4819135        -0.4502914           0.396500
##   pctFGPerGameTeam
## 0        0.4599476
## 1       -0.4337160
##
## Coefficients of linear discriminants:
##                                  LD1
## Ranking                   2.14112425
## winsTeam                  0.26375642
## marginVictoryTeam        -3.81276206
## nrtgTeamMisc              3.69726412
## drtgTeamMisc              0.15910554
## pctFGPerGameOpponent      0.00836316
## ortgTeamMisc              0.08684611
## pctEFGTeamOppMisc         0.01004129
## pctTrueShootingeTeamMisc -0.31375616
## pctFG2PerGameOpponent    -0.05629417
## pctEFGTeamMisc            0.42667420
## pctFG2PerGameTeam         0.08775697
## ptsPerGameOpponent       -0.23368187
## pctFGPerGameTeam         -0.09771885
```

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

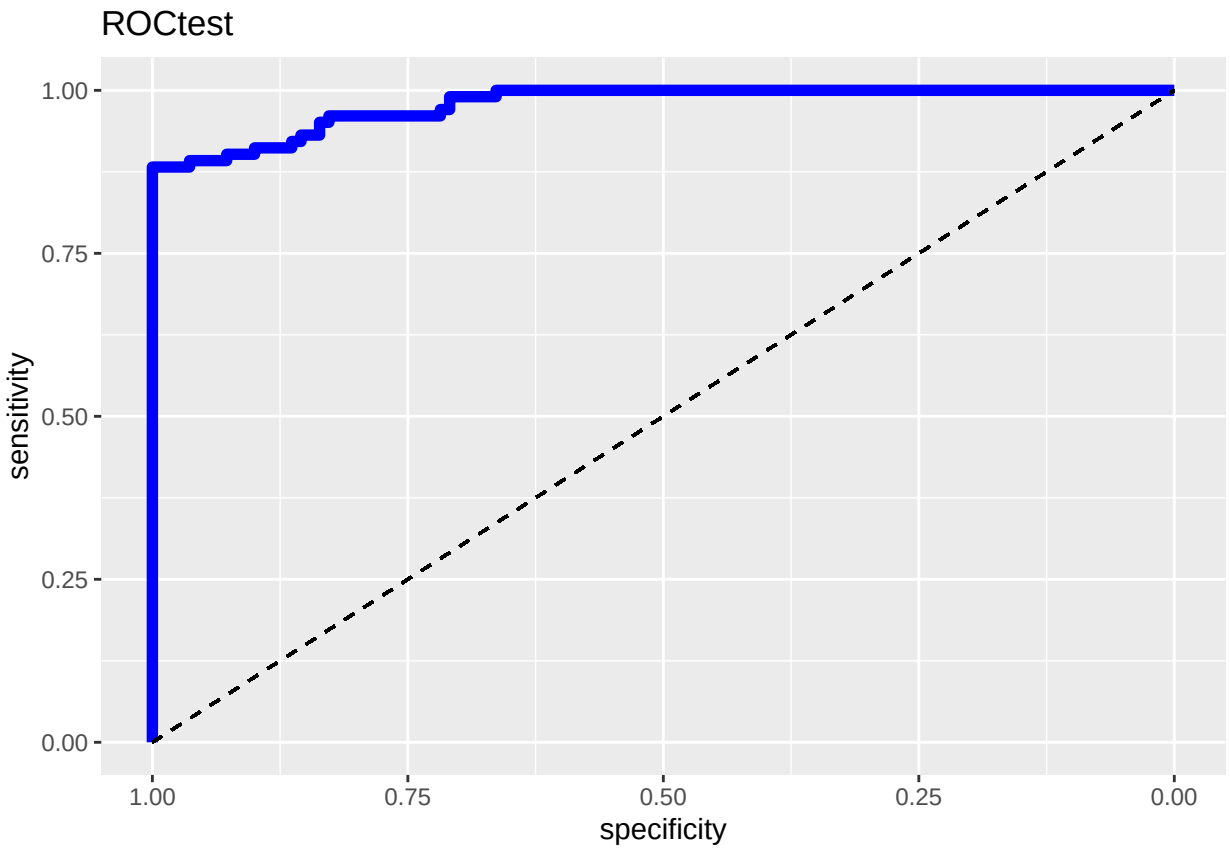
```
##       y_pred
## y_true  0  1
##      0 99 11
##      1  9 93
```

```
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 + marginVictoryTeam +
                    nrtgTeamMisc + drtgTeamMisc +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    pctEFGTeamOppMisc + pctTrueShootingeTeamMisc +
                    pctFG2PerGameOpponent + pctEFGTeamMisc +
                    pctFG2PerGameTeam + ageMeanMisc +
                    astPerGameOpponent + drbPerGameTeam +
                    fgmPerGameOpponent + 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
##      0 101   9
##      1  12  90
```

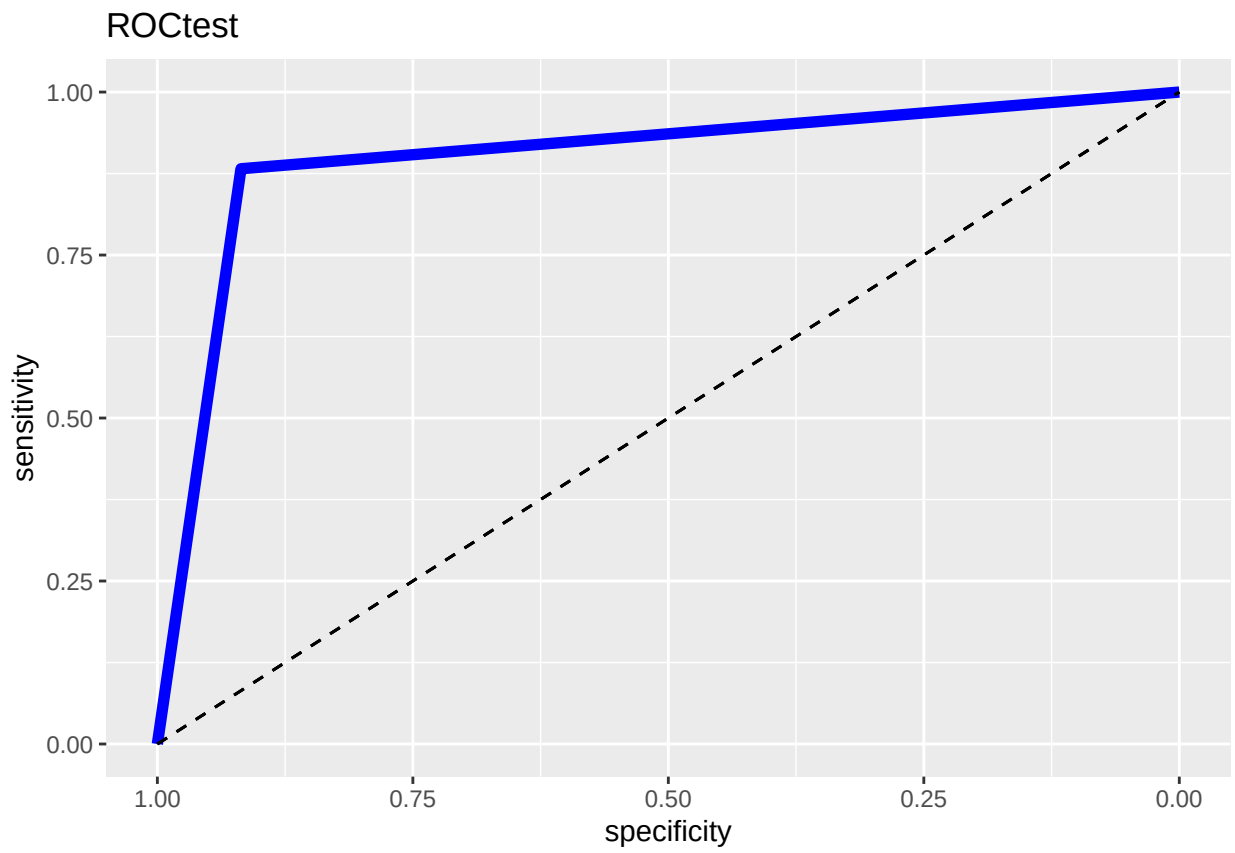
```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datat
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$ytest, knntest)),0,Precision(datatest$y
Recallknn <- Recallknn + ifelse(is.nan(Recall(datatest$ytest, knntest)),0,Recall(datatest$ytest, 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] 0.08155236
```

```
Accuracyglm/4
```

```
## [1] 0.9184476
```

```
Precisionglm/4
```

```
## [1] 0.9081982
```

```
Recallglm/4
```

```
## [1] 0.9316275
```

```
F1glm/4
```

```
## [1] 0.9194962
```

```
AUCglm/4
```

```
## [1] 0.9186381
```

```
ConfusMatglm
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0  94   6
##      1   6 105
##
## [[2]]
##       y_pred
## y_true  0  1
##      0 99 10
##      1 11 91
##
## [[3]]
##       y_pred
## y_true  0  1
##      0 97  5
##      1 14 96
##
## [[4]]
##       y_pred
## y_true   0   1
##      0 102   8
##      1   9  93
```

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

```
## [1] 1.054346
```

```
Accuracylda/4
```

```
## [1] 0.9220245
```

```
Precisionlda/4
```

```
## [1] 0.9121764
```

```
Recalllda/4
```

```
## [1] 0.9342391
```

```
F1lda/4
```

```
## [1] 0.9228092
```

```
AUClda/4
```

```
## [1] 0.9224942
```

```
ConfusMatlda
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0  96   4
##      1   7 104
##
## [[2]]
##       y_pred
## y_true   0   1
##      0 102   7
##      1   8  94
##
## [[3]]
##       y_pred
## y_true  0  1
##      0 96  6
##      1 14 96
##
## [[4]]
##       y_pred
## y_true  0  1
##      0 99 11
##      1  9 93
```

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

```
## [1] 0.1111173
```

```
Accuracyknn/4
```

```
## [1] 0.8888827
```

```
Precisionknn/4
```

```
## [1] 0.8686412
```

```
Recallknn/4
```

```
## [1] 0.9143444
```

```
F1knn/4
```

```
## [1] 0.8908118
```

```
AUCknn/4
```

```
## [1] 0.889209
```

```
ConfusMatknn
```

```
## [[1]]
##       y_pred
## y_true  0  1
##      0 91  9
##      1 17 94
##
## [[2]]
##       y_pred
## y_true   0   1
##      0 100   9
##      1  13  89
##
## [[3]]
##       y_pred
## y_true  0  1
##      0 93  9
##      1 16 94
##
## [[4]]
##       y_pred
## y_true   0   1
##      0 101   9
##      1  12  90
```

```
##2020 Season Predictions
ytrain <- ceiling((MasterPerGame$finish-4)/5)
xtrain <- MasterPerGame[,-3]
datatrain <- cbind(ytrain, xtrain)
xtest <- MasterPerGame2020

##Logistic Regression
model.glm <- glm(ytrain~ Ranking + blkPerGameTeam + pctEFGTeamMisc +
                   nrtgTeamMisc + drtgTeamMisc + pctFGPerGameTeam +
                   ortgTeamMisc + pctTrueShootingeTeamMisc +
                   marginVictoryTeam + pctEFGTeamOppMisc + winsTeam +
                   stlPerGameTeam + fg2aPerGameOpponent + fg2mPerGameOpponent,
                 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  0  1  1  1  1  0  1  1  0  0  0  0  1  0  0  1  1  1  0  1  0  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] "Brooklyn Nets"
## [1] "Denver Nuggets"
## [1] "Houston Rockets"
## [1] "Indiana Pacers"
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Miami Heat"
## [1] "Milwaukee Bucks"
## [1] "Oklahoma City Thunder"
## [1] "Philadelphia 76ers"
## [1] "Toronto Raptors"
## [1] "Utah Jazz"
```

```
##Logistic Regression Model suggests the Boston Celtics, Brooklyn Nets, Denver Nuggets,
##Houston Rockets, Indiana Pacers, LA Clippers, Los Angeles Lakers, Miami Heat,
##Milwaukee Bucks, Oklahoma City Thunder, Philadelphia 76ers, Toronto Raptors
```

```r
##and Utah Jazz are Playoff level teams

##Discriminant Models
##Linear Discriminant
model.lda <- lda(ytrain~ Ranking + winsTeam + marginVictoryTeam +
                   nrtgTeamMisc + drtgTeamMisc + pctFGPerGameOpponent +
                   ortgTeamMisc + pctEFGTeamOppMisc + pctTrueShootingeTeamMisc +
                   pctFG2PerGameOpponent + pctEFGTeamMisc + pctFG2PerGameTeam +
                   ptsPerGameOpponent + pctFGPerGameTeam, data = datatrain)
model.lda
```

```
## Call:
## lda(ytrain ~ Ranking + winsTeam + marginVictoryTeam + nrtgTeamMisc +
##     drtgTeamMisc + pctFGPerGameOpponent + ortgTeamMisc + pctEFGTeamOppMisc +
##     pctTrueShootingeTeamMisc + pctFG2PerGameOpponent + pctEFGTeamMisc +
##     pctFG2PerGameTeam + ptsPerGameOpponent + pctFGPerGameTeam,
##     data = datatrain)
##
## Prior probabilities of groups:
##         0         1
## 0.4810875 0.5189125
##
## Group means:
##      Ranking   winsTeam marginVictoryTeam nrtgTeamMisc drtgTeamMisc
## 0 -0.8539531  0.8043156         0.7759869    0.7760598   -0.6334965
## 1  0.7917059 -0.7456866        -0.7194229   -0.7194905    0.5873191
##   pctFGPerGameOpponent ortgTeamMisc pctEFGTeamOppMisc pctTrueShootingeTeamMisc
## 0           -0.5815899    0.5740793        -0.5761377                0.5401231
## 1            0.5391961   -0.5322330         0.5341413               -0.5007519
##   pctFG2PerGameOpponent pctEFGTeamMisc pctFG2PerGameTeam ptsPerGameOpponent
## 0            -0.5395989      0.5222692         0.5102704         -0.4869573
## 1             0.5002659     -0.4841995        -0.4730753          0.4514616
##   pctFGPerGameTeam
## 0        0.4934952
## 1       -0.4575229
##
## Coefficients of linear discriminants:
##                                 LD1
## Ranking                  1.89508975
## winsTeam                 0.13089053
## marginVictoryTeam       -1.35077812
## nrtgTeamMisc             1.11066131
## drtgTeamMisc             0.05016322
## pctFGPerGameOpponent    -0.08068486
## ortgTeamMisc             0.21615772
## pctEFGTeamOppMisc        0.10635303
## pctTrueShootingeTeamMisc -0.39858007
## pctFG2PerGameOpponent    -0.10843722
## pctEFGTeamMisc            0.69299179
## pctFG2PerGameTeam        -0.29644953
## ptsPerGameOpponent       -0.09512508
## pctFGPerGameTeam         -0.01805743
```

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

```
##  [1] 1 0 0 1 1 1 1 0 1 1 0 0 0 0 1 0 0 1 1 1 0 0 0 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] "Brooklyn Nets"
## [1] "Denver Nuggets"
## [1] "Houston Rockets"
## [1] "Indiana Pacers"
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Miami Heat"
## [1] "Milwaukee Bucks"
## [1] "Oklahoma City Thunder"
## [1] "Orlando Magic"
## [1] "Philadelphia 76ers"
## [1] "Toronto Raptors"
## [1] "Utah Jazz"
```

```
##Discriminant Analysis Model suggests the Boston Celtics, Brooklyn Nets, Denver Nuggets,
##Houston Rockets, Indiana Pacers, LA Clippers, Los Angeles Lakers, Miami Heat,
##Milwaukee Bucks, Oklahoma City Thunder, Orlando Magic, Philadelphia 76ers,
##Toronto Raptors and Utah Jazz are Playoff level teams

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

```
##  [1] 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 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] "Brooklyn Nets"
## [1] "Dallas Mavericks"
## [1] "Denver Nuggets"
## [1] "Houston Rockets"
## [1] "Indiana Pacers"
## [1] "Los Angeles Clippers"
## [1] "Los Angeles Lakers"
## [1] "Miami Heat"
## [1] "Milwaukee Bucks"
## [1] "Oklahoma City Thunder"
## [1] "Philadelphia 76ers"
## [1] "Toronto Raptors"
## [1] "Utah Jazz"
```

```
##K Nearest Neighbours that the Boston Celtics, Brooklyn Nets, Dallas Mavericks,
##Denver Nuggets, Houston Rockets, Indiana Pacers, LA Clippers, Los Angeles Lakers,
##Miami Heat, Milwaukee Bucks, Oklahoma City Thunder, Philadelphia 76ers,
##Toronto Raptors and Utah Jazz are Playoff Level Teams
```