NBAAnalysisChampionClassificationModel.R

dpesl

2020-05-17

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

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

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

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

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

```r
##this is done because we want to avoid running into problems with
##changes in game plans (we will see whether teams are better at 3pts compared
##to league in a paricular season, vs over 29 seasons)
##then we re-scale all together
MasterPerGame2020[,-c((1:5),8)] <- scale(MasterPerGame2020[,-c((1:5),8)])
MasterPerGame[(1:27),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(1:27),-c((1:5),7,(9:10))])
MasterPerGame[(28:54),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(28:54),-c((1:5),7,(9:10))])
MasterPerGame[(55:81),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(55:81),-c((1:5),7,(9:10))])
MasterPerGame[(82:108),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(82:108),-c((1:5),7,(9:10))])
MasterPerGame[(109:135),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(109:135),-c((1:5),7,(9:10))])
MasterPerGame[(136:164),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(136:164),-c((1:5),7,(9:10))])
MasterPerGame[(165:193),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(165:193),-c((1:5),7,(9:10))])
MasterPerGame[(194:222),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(194:222),-c((1:5),7,(9:10))])
MasterPerGame[(223:251),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(223:251),-c((1:5),7,(9:10))])
MasterPerGame[(252:280),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(252:280),-c((1:5),7,(9:10))])
MasterPerGame[(281:309),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(281:309),-c((1:5),7,(9:10))])
MasterPerGame[(310:338),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(310:338),-c((1:5),7,(9:10))])
MasterPerGame[(339:367),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(339:367),-c((1:5),7,(9:10))])
MasterPerGame[(368:396),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(368:396),-c((1:5),7,(9:10))])
MasterPerGame[(397:426),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(397:426),-c((1:5),7,(9:10))])
MasterPerGame[(427:456),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(427:456),-c((1:5),7,(9:10))])
MasterPerGame[(457:486),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(457:486),-c((1:5),7,(9:10))])
MasterPerGame[(487:516),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(487:516),-c((1:5),7,(9:10))])
MasterPerGame[(517:546),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(517:546),-c((1:5),7,(9:10))])
MasterPerGame[(547:576),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(547:576),-c((1:5),7,(9:10))])
MasterPerGame[(577:606),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(577:606),-c((1:5),7,(9:10))])
MasterPerGame[(607:636),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(607:636),-c((1:5),7,(9:10))])
MasterPerGame[(637:666),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(637:666),-c((1:5),7,(9:10))])
MasterPerGame[(667:696),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(667:696),-c((1:5),7,(9:10))])
MasterPerGame[(697:726),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(697:726),-c((1:5),7,(9:10))])
MasterPerGame[(727:756),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(727:756),-c((1:5),7,(9:10))])
MasterPerGame[(757:786),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(757:786),-c((1:5),7,(9:10))])
MasterPerGame[(787:816),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(787:816),-c((1:5),7,(9:10))])
MasterPerGame[(817:846),-c((1:5),7,(9:10))] <- scale(MasterPerGame[(817:846),-c((1:5),7,(9:10))])
MasterPerGame2020[,-c((1:5),8)] <- (MasterPerGame2020[,-c((1:5),8)] - colMeans(MasterPerGame[,-c((1:5),7
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,]


ytrain <- ceiling(MasterPerGame$finish/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
```

```
## Loading required package: ggplot2
```

```
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: algorithm did not converge
```

```
## 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
```

```
## 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
```

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

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

```
## Warning: glm.fit: algorithm did not converge
```

```
## 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
```

```
## 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
```

```
## 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
## Warning: glm.fit: algorithm did not converge
```

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

```
## Warning: glm.fit: algorithm did not converge
```

```
## 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   0.9622 0.08271   0.006597 0.09723        *
##          6   0.9622 0.08271   0.006597 0.09723
##          7   0.9610 0.11729   0.008887 0.16725
##          8   0.9610 0.08062   0.005872 0.09966
##          9   0.9586 0.07275   0.004423 0.09407
##         10   0.9586 0.07275   0.004423 0.09407
##         11   0.9539 0.10688   0.007965 0.08615
##         12   0.9551 0.14442   0.006050 0.12820
##         13   0.9539 0.14101   0.008006 0.12899
##         14   0.9539 0.14101   0.008006 0.12899
##         15   0.9551 0.18204   0.009775 0.16749
##         16   0.9563 0.20754   0.009660 0.20068
##         17   0.9575 0.24092   0.013836 0.26671
##         18   0.9575 0.29404   0.016759 0.25707
##         19   0.9586 0.29570   0.016915 0.25443
##         20   0.9586 0.29581   0.016906 0.25068
##         21   0.9586 0.29570   0.016915 0.25443
##         22   0.9575 0.28549   0.015831 0.24092
##         23   0.9563 0.28216   0.018168 0.24278
##         24   0.9516 0.27343   0.022210 0.25100
##         25   0.9516 0.27228   0.024725 0.25058
##         62   0.9303 0.20266   0.010493 0.06934
##
## The top 5 variables (out of 5):
##    pctTOVOpponentMisc, pctDRBOpponentMisc, paceTeamMisc, fg3mPerGameTeam, trbPerGameOpponent
```

```
model.glm$optVariables
```

```
## [1] "pctTOVOpponentMisc" "pctDRBOpponentMisc" "paceTeamMisc"
## [4] "fg3mPerGameTeam"    "trbPerGameOpponent"
```

```r
##Discriminant Analysis Feature Selection
##Linear Discriminant
##install.packages("MASS")
library(MASS)
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   0.9669 0.05391  9.035e-05 0.10782
##          6   0.9657 0.05182  2.423e-03 0.10929
##          7   0.9657 0.05182  2.423e-03 0.10929
##          8   0.9645 0.04484  2.732e-03 0.09535
##          9   0.9657 0.04693  2.308e-03 0.09387
##         10   0.9645 0.04484  2.732e-03 0.09535
##         11   0.9645 0.04484  2.732e-03 0.09535
##         12   0.9645 0.10413  6.116e-03 0.13103
##         13   0.9693 0.17578  2.646e-03 0.11793        *
##         14   0.9645 0.09750  2.633e-03 0.11781
##         15   0.9645 0.09750  2.633e-03 0.11781
##         16   0.9657 0.15843  4.444e-03 0.11352
##         17   0.9657 0.15843  4.444e-03 0.11352
##         18   0.9657 0.15843  4.444e-03 0.11352
##         19   0.9645 0.09750  2.633e-03 0.11781
##         20   0.9634 0.09586  4.435e-03 0.11980
##         21   0.9645 0.14554  2.633e-03 0.09774
##         22   0.9622 0.13329  3.871e-03 0.08960
##         23   0.9634 0.13883  2.312e-03 0.09257
##         24   0.9645 0.14553  2.732e-03 0.09774
##         25   0.9645 0.14553  2.732e-03 0.09774
##         62   0.9610 0.16628  9.014e-03 0.21119
##
## The top 5 variables (out of 13):
##    winsTeam, nrtgTeamMisc, marginVictoryTeam, Ranking, drtgTeamMisc
```

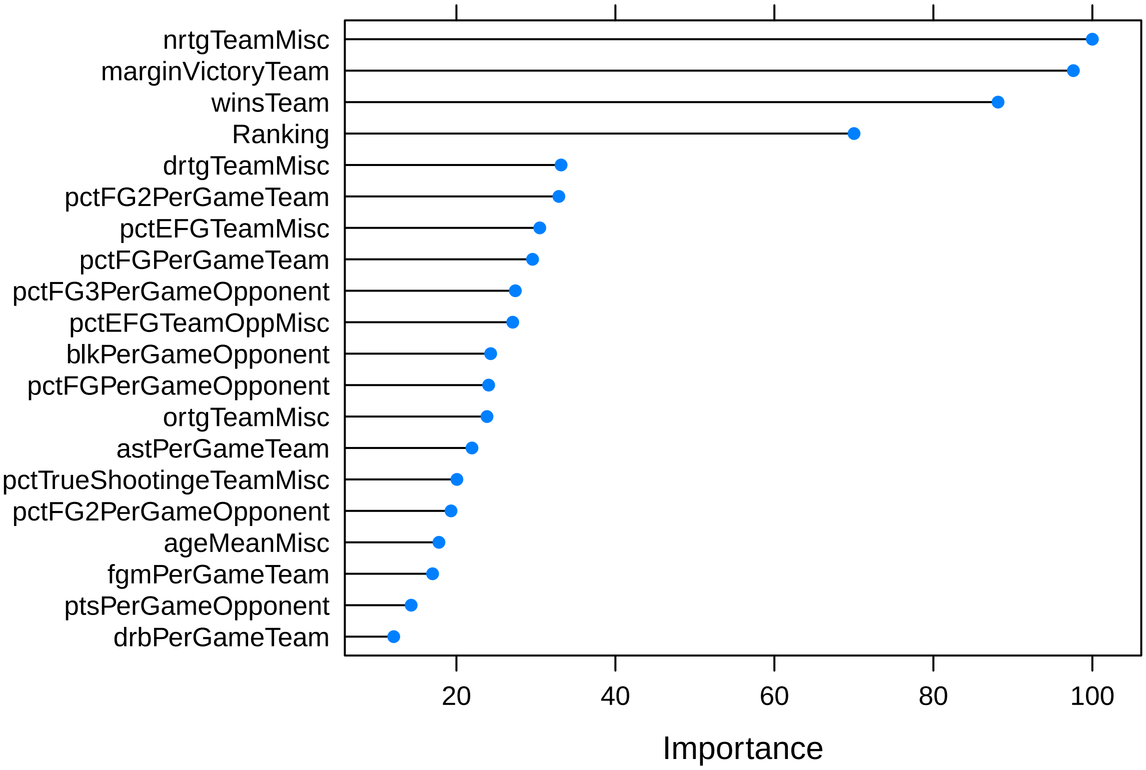
```r
model.lda$optVariables
```

```
##  [1] "winsTeam"             "nrtgTeamMisc"         "marginVictoryTeam"
##  [4] "Ranking"              "drtgTeamMisc"         "pctFG2PerGameTeam"
##  [7] "pctFGPerGameTeam"     "pctFGPerGameOpponent" "pctEFGTeamOppMisc"
## [10] "pctEFGTeamMisc"       "ortgTeamMisc"         "pctFG2PerGameOpponent"
## [13] "blkPerGameOpponent"
```

```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
## nrtgTeamMisc                100.00
## marginVictoryTeam            97.61
## winsTeam                     88.14
## Ranking                      70.02
## drtgTeamMisc                 33.17
## pctFG2PerGameTeam            32.89
## pctEFGTeamMisc               30.49
## pctFGPerGameTeam             29.58
## pctFG3PerGameOpponent        27.42
## pctEFGTeamOppMisc            27.09
## blkPerGameOpponent           24.31
## pctFGPerGameOpponent         24.06
## ortgTeamMisc                 23.85
## astPerGameTeam               21.96
## pctTrueShootingeTeamMisc     20.06
## pctFG2PerGameOpponent        19.32
## ageMeanMisc                  17.80
## fgmPerGameTeam               17.00
## ptsPerGameOpponent           14.31
## drbPerGameTeam               12.11
```

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



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

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

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

##
## Attaching package: 'MLmetrics'

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

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

```
##Championship Analysis
##1st fold = validation set
MSEglm <- 0
Accuracyglm <- 0
```

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

##Logistic Regression
##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
model.glm <- glm(ytrain~pctTOVOpponentMisc + pctDRBOpponentMisc +
                   paceTeamMisc + fg3mPerGameTeam + trbPerGameOpponent,
                 data = datatrain, family = binomial)
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- as.factor(0)
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- as.factor(1)
  }
```

```
}
ConfusMatglm[[1]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[1]]
```

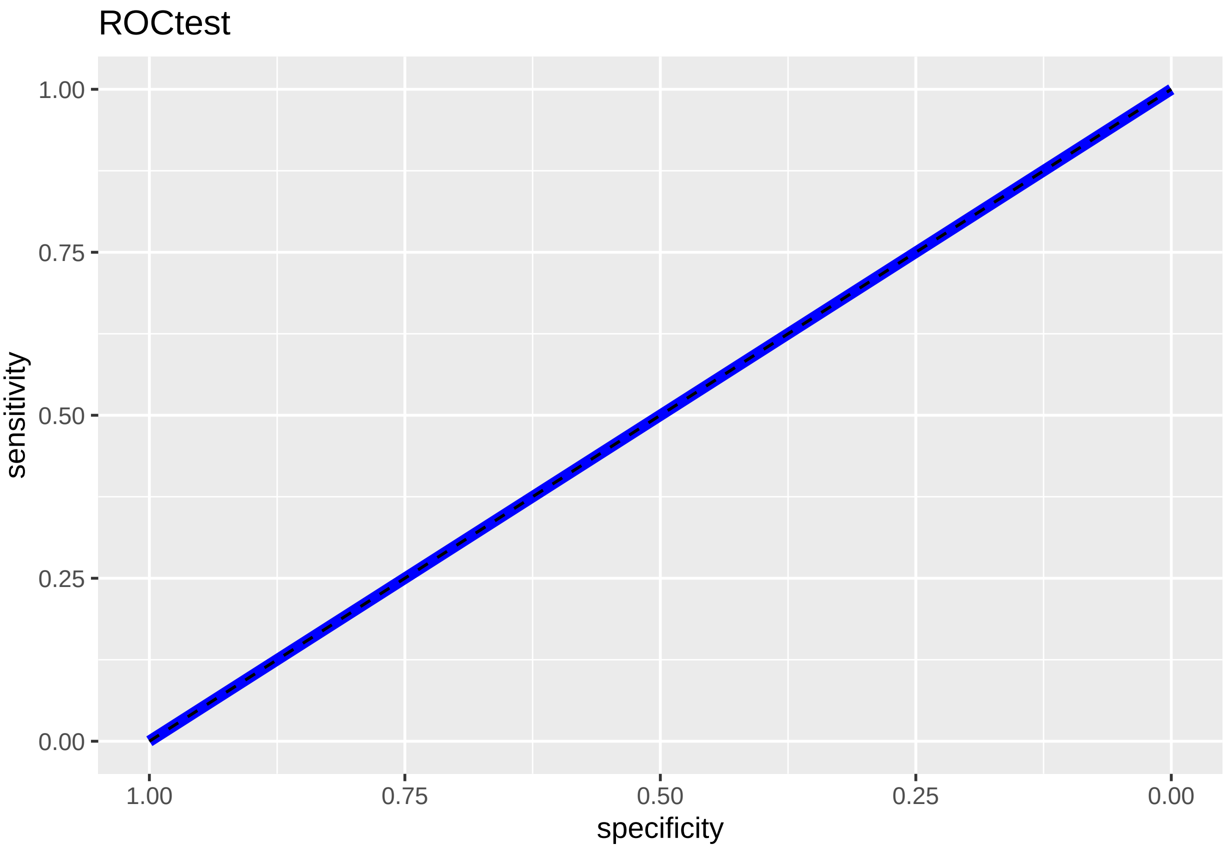
```
##        y_pred
## y_true   0   1
##      0   0   4
##      1   0 207
```

```
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$ytest):max(dat
F1glm <- F1glm + ifelse(is.nan(F1_Score(factor(datatest$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~ 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
## 0.0503937 0.9496063
##
## Group means:
##      winsTeam nrtgTeamMisc marginVictoryTeam     Ranking drtgTeamMisc
## 0  1.51059582   1.52675158        1.51426553 -1.38351575  -1.20236338
## 1 -0.04411622  -0.04588031       -0.04547423  0.02513471   0.02316409
##   pctFG2PerGameTeam pctFGPerGameTeam pctEFGTeamOppMisc pctFGPerGameOpponent
## 0        1.27494847       1.10167439      -1.146123768         -1.084008639
## 1       -0.04523581      -0.04157287       0.002649937          0.003298623
##   pctEFGTeamMisc ortgTeamMisc blkPerGameOpponent pctFG2PerGameOpponent
## 0     1.06638085   1.14940611        -1.18768650           -1.01761012
## 1    -0.03968859  -0.04592527         0.07304191           -0.01492096
##
## Coefficients of linear discriminants:
##                              LD1
## winsTeam              -0.2566179
## nrtgTeamMisc          -9.9132710
## marginVictoryTeam      8.7947248
## Ranking               -0.4364967
## drtgTeamMisc          -0.2441597
## pctFG2PerGameTeam     -2.0005472
## pctFGPerGameTeam       0.4994304
## pctEFGTeamOppMisc      1.1250062
## pctFGPerGameOpponent  -0.3941482
## pctEFGTeamMisc         1.4484760
## ortgTeamMisc           0.2325039
## blkPerGameOpponent     0.3797272
## pctFG2PerGameOpponent -0.6808489
```

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

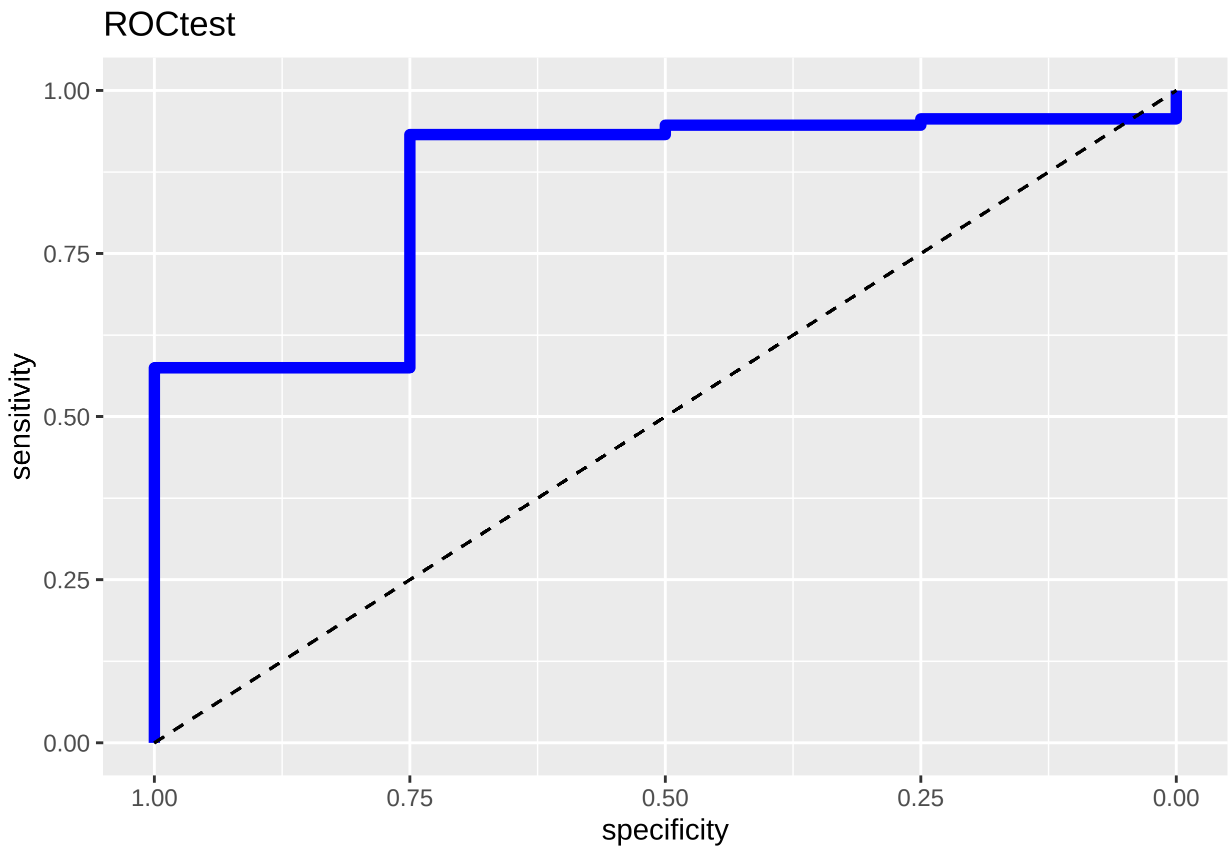
```
##       y_pred
## y_true   0   1
##      0   0   4
##      1   2 205
```

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

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

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

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



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

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

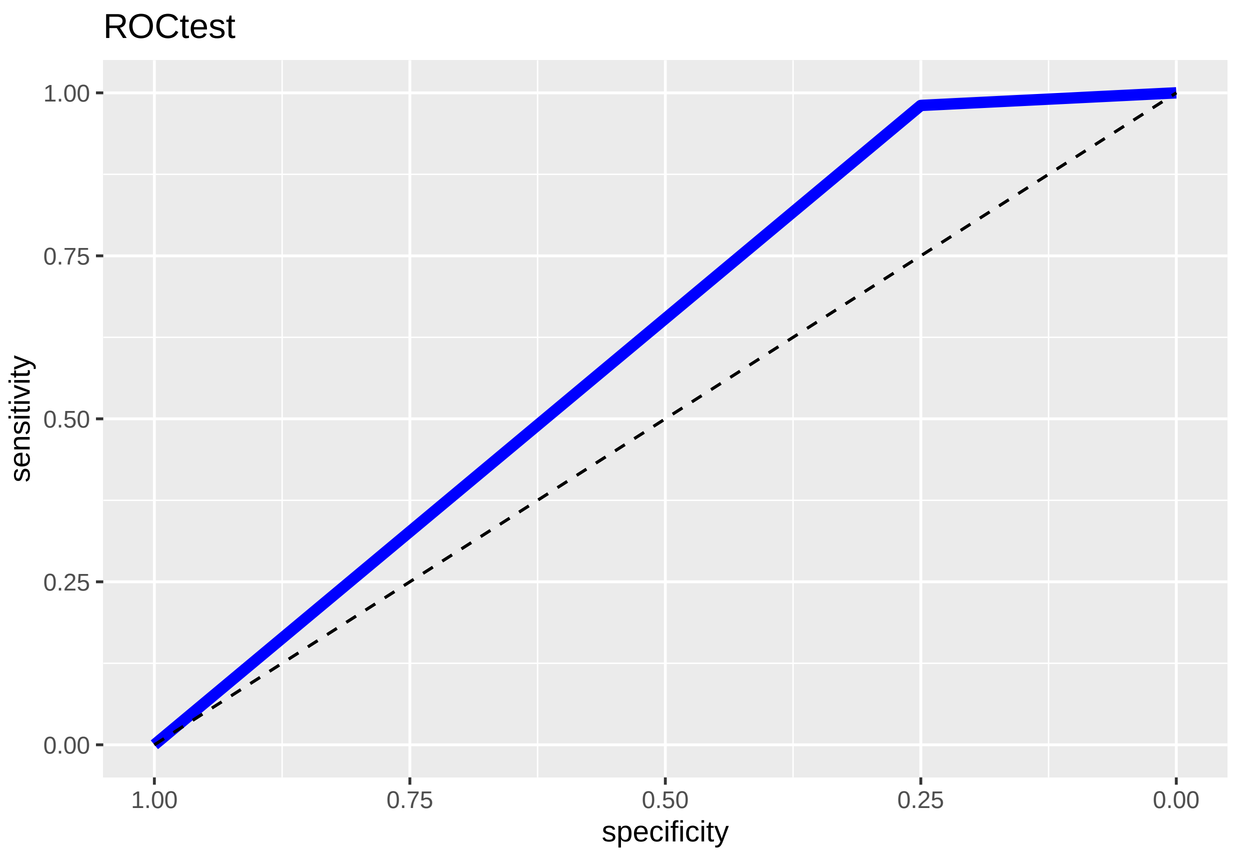
```
##       y_pred
## y_true   0   1
##      0   1   3
##      1   4 203
```

```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datate
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$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()
```



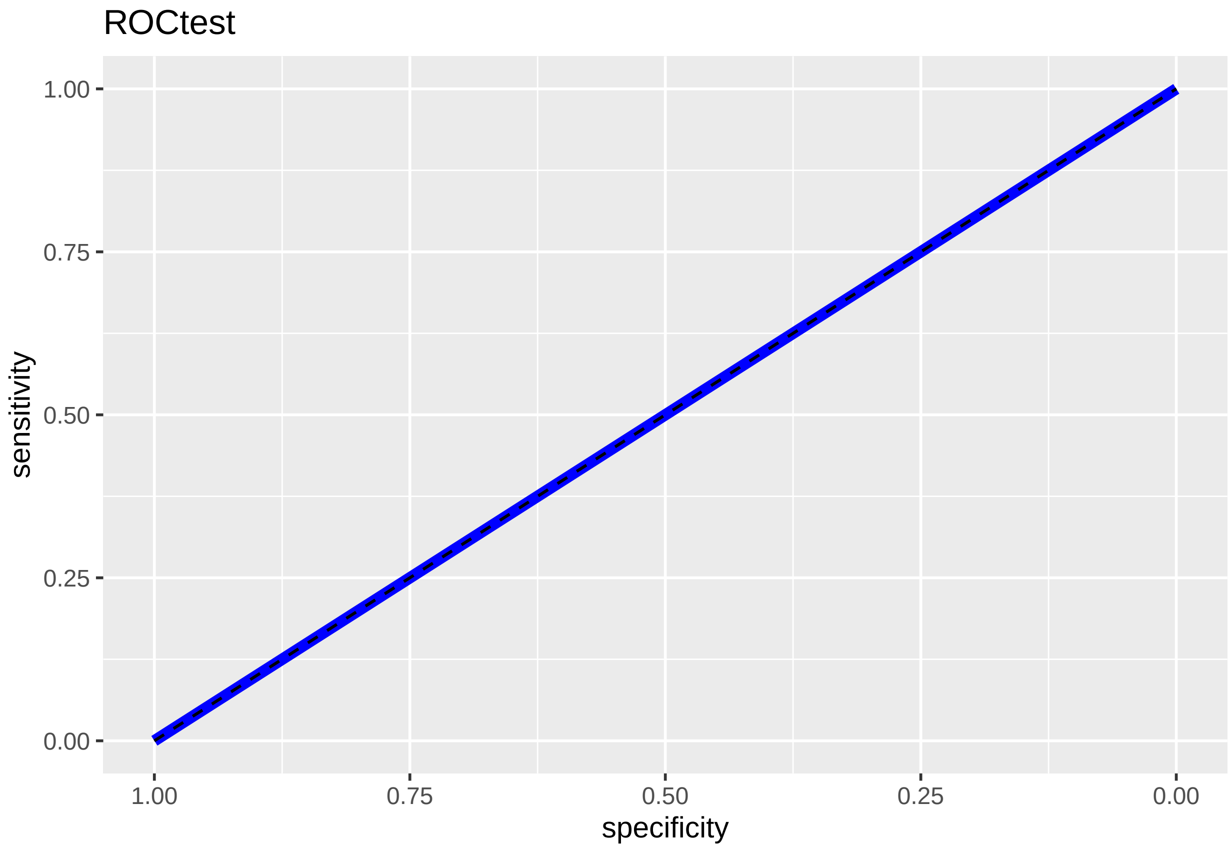
```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

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

##Logistic Regression
model.glm <- glm(ytrain~pctTOVOpponentMisc + pctDRBOpponentMisc +
                   paceTeamMisc + fg3mPerGameTeam + trbPerGameOpponent,
                 data = datatrain, family = binomial)
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- as.factor(0)
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- as.factor(1)
  }
}
ConfusMatglm[[2]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[2]]
```

```
##       y_pred
## y_true   0   1
##      0   0  12
##      1   0 199
```

```
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~ 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
## 0.03779528 0.96220472
##
## Group means:
##      winsTeam nrtgTeamMisc marginVictoryTeam    Ranking drtgTeamMisc
## 0  1.45556296   1.39386187        1.38482042 -1.3620877 -1.097523229
## 1 -0.03779519  -0.02909832       -0.02919484  0.0274381  0.005170199
##   pctFG2PerGameTeam pctFGPerGameTeam pctEFGTeamOppMisc pctFGPerGameOpponent
## 0        1.32425937       1.09664006      -1.104267437         -1.059143852
## 1       -0.05776121      -0.05416965       0.003191722          0.005466309
##   pctEFGTeamMisc ortgTeamMisc blkPerGameOpponent pctFG2PerGameOpponent
## 0     1.12582321   1.06183098        -1.25759398          -1.075108156
## 1    -0.05162779  -0.03304898         0.02356176          -0.007893437
##
## Coefficients of linear discriminants:
##                               LD1
## winsTeam              -0.99383725
## nrtgTeamMisc          -8.16819708
## marginVictoryTeam      8.68913007
## Ranking               -0.19376328
## drtgTeamMisc           0.13756936
## pctFG2PerGameTeam     -1.80295716
## pctFGPerGameTeam       0.54156200
## pctEFGTeamOppMisc      0.51338194
## pctFGPerGameOpponent  -1.00553623
## pctEFGTeamMisc         1.02774786
## ortgTeamMisc          -0.08526521
## blkPerGameOpponent     0.35637570
## pctFG2PerGameOpponent  0.58916242
```

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

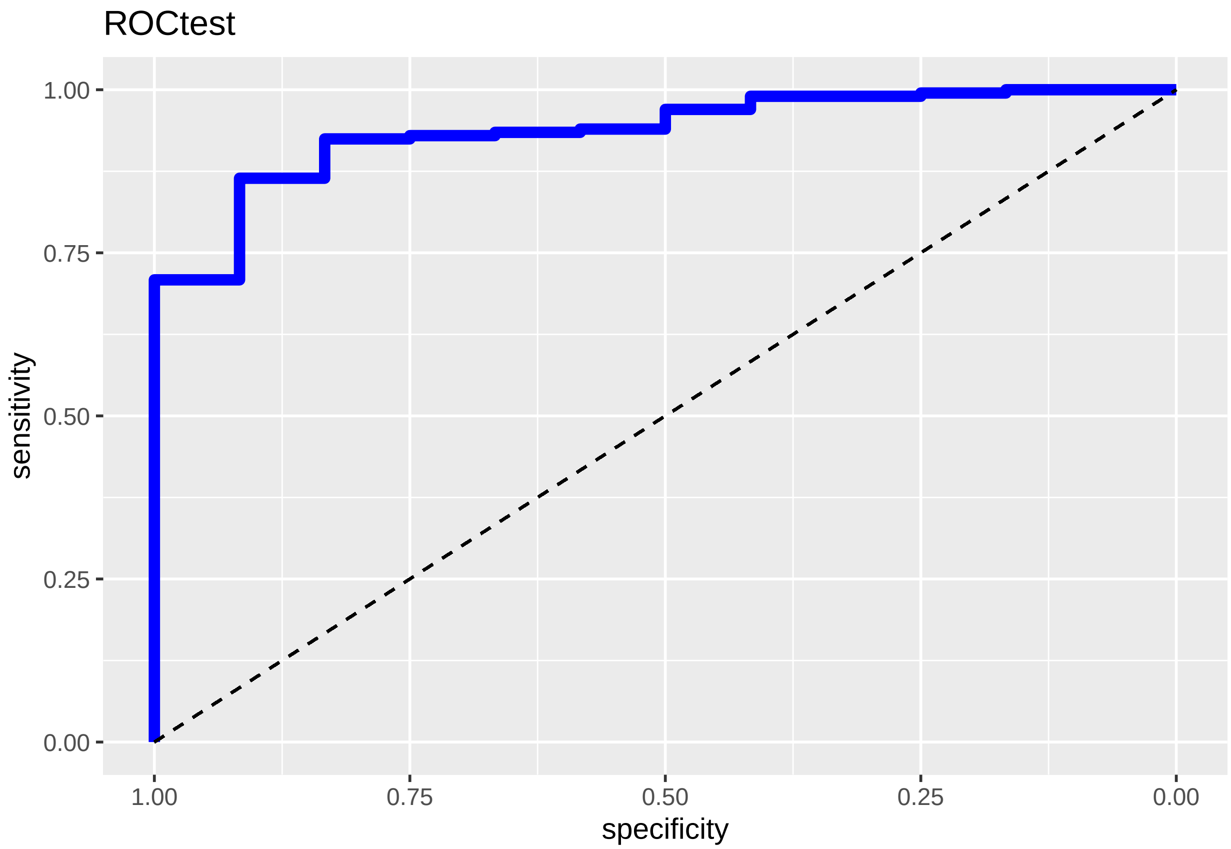
```
##       y_pred
## y_true   0   1
##      0   2  10
##      1   0 199
```

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

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

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

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



```
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)~ nrtgTeamMisc + marginVictoryTeam +
                    winsTeam + Ranking + drtgTeamMisc +
                    pctFG2PerGameTeam + pctEFGTeamMisc +
                    pctFGPerGameTeam + pctFG3PerGameOpponent +
                    pctEFGTeamOppMisc + blkPerGameOpponent +
                    pctFGPerGameOpponent + ortgTeamMisc +
                    astPerGameTeam + pctTrueShootingeTeamMisc,
                  data = datatrain, k = knnval)
knntest <- predict(model.knn, datatest, type = "class")
ConfusMatknn[[2]] <- ConfusionMatrix(knntest, datatest$ytest)
ConfusMatknn[[2]]
```

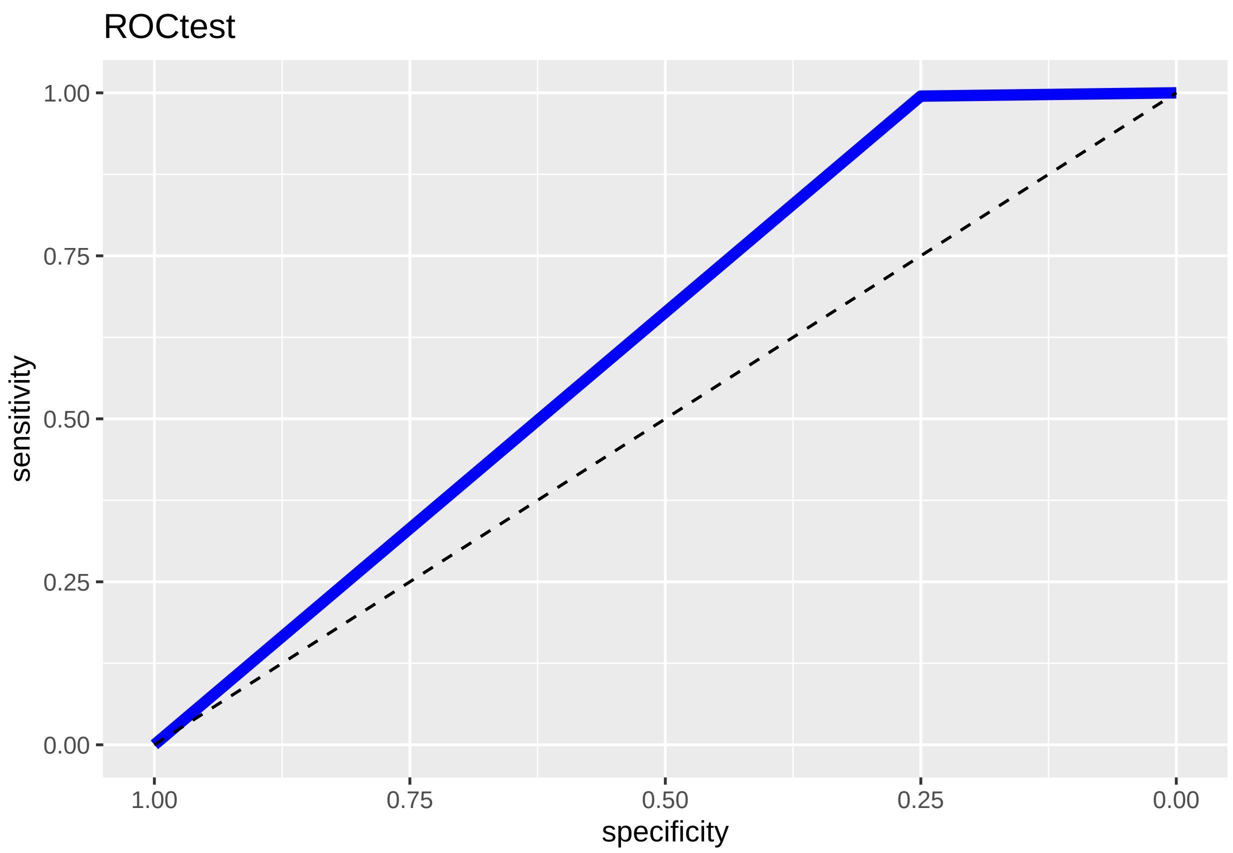
```
##         y_pred
## y_true   0   1
##      0   3   9
##      1   1 198
```

```
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()
```



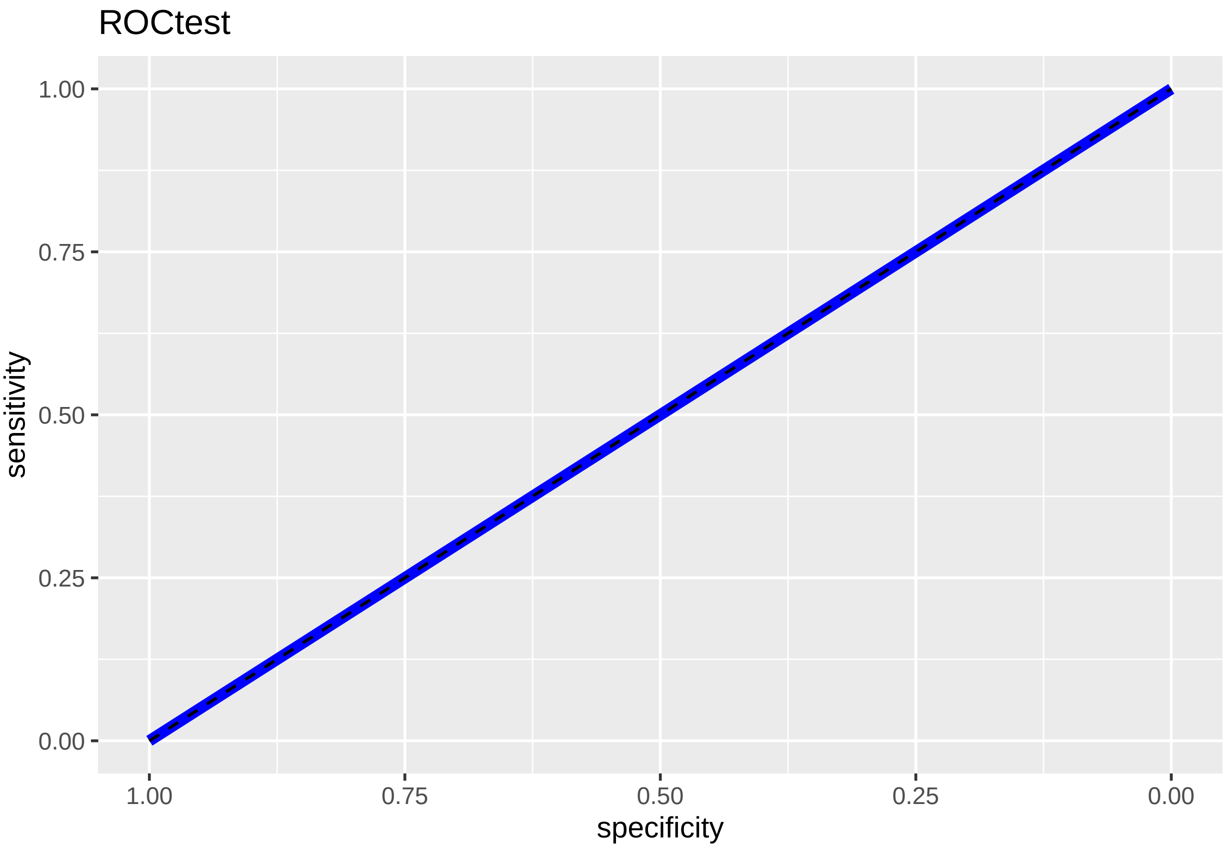
```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

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

##Logistic Regression
model.glm <- glm(ytrain~pctTOVOpponentMisc + pctDRBOpponentMisc +
                   paceTeamMisc + fg3mPerGameTeam + trbPerGameOpponent,
                 data = datatrain, family = binomial)
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- as.factor(0)
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- as.factor(1)
  }
}
ConfusMatglm[[3]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[3]]
```

```
##       y_pred
## y_true   0   1
##      0   0  12
##      1   0 200
```

```
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~ 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
## 0.03785489 0.96214511
##
## Group means:
##      winsTeam nrtgTeamMisc marginVictoryTeam     Ranking drtgTeamMisc
## 0  1.53332319  1.523098026       1.514585555 -1.40077440 -1.171996626
## 1 -0.01983453 -0.009205824      -0.008385834  0.01228686  0.008037639
##   pctFG2PerGameTeam pctFGPerGameTeam pctEFGTeamOppMisc pctFGPerGameOpponent
## 0       1.326972129        1.1698868      -1.088309346         -1.048336962
## 1      -0.008022794       -0.0110779      -0.009976638         -0.008666434
##   pctEFGTeamMisc ortgTeamMisc blkPerGameOpponent pctFG2PerGameOpponent
## 0     1.12932118  1.186315269        -1.28268753           -0.95392523
## 1    -0.02037176 -0.005900971         0.03110604           -0.01650789
##
## Coefficients of linear discriminants:
##                               LD1
## winsTeam              -0.71100178
## nrtgTeamMisc          -5.52165367
## marginVictoryTeam      5.32759065
## Ranking               -0.38139481
## drtgTeamMisc           0.12984893
## pctFG2PerGameTeam     -1.66480188
## pctFGPerGameTeam       0.24317231
## pctEFGTeamOppMisc      0.63681680
## pctFGPerGameOpponent  -0.03207366
## pctEFGTeamMisc         1.37621307
## ortgTeamMisc          -0.04207164
## blkPerGameOpponent     0.42324622
## pctFG2PerGameOpponent -0.64590073
```

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

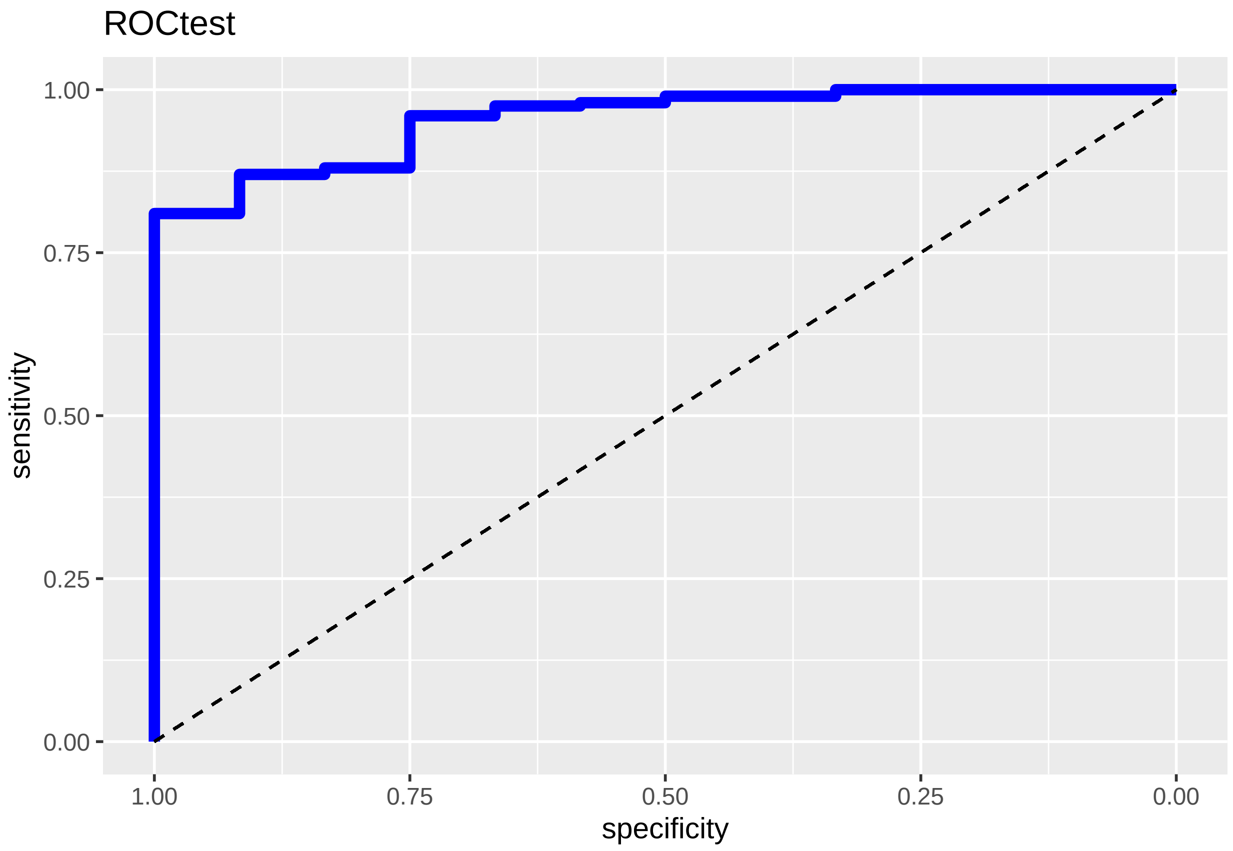
```
##       y_pred
## y_true   0   1
##      0   3   9
##      1   0 200
```

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

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

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

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



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

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

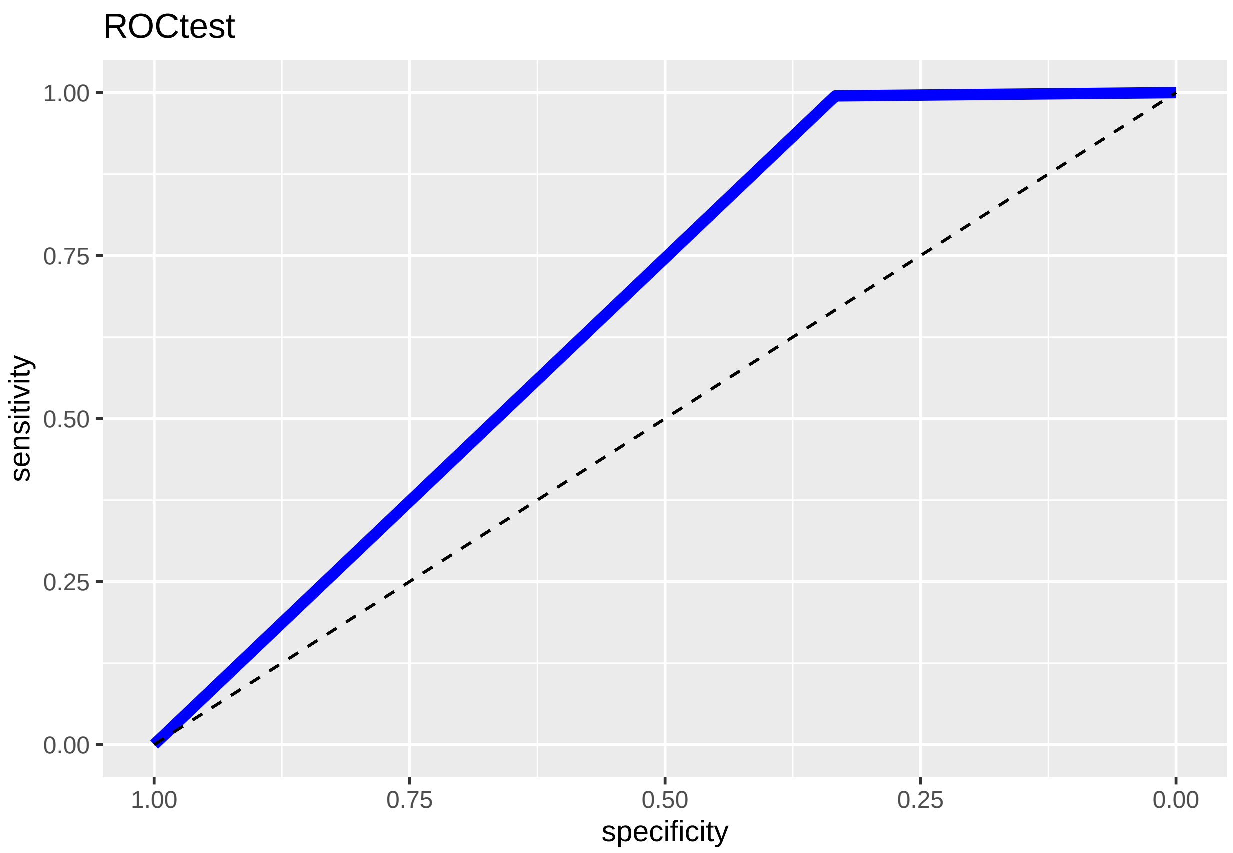
```
##       y_pred
## y_true   0   1
##      0   4   8
##      1   1 199
```

```
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()
```



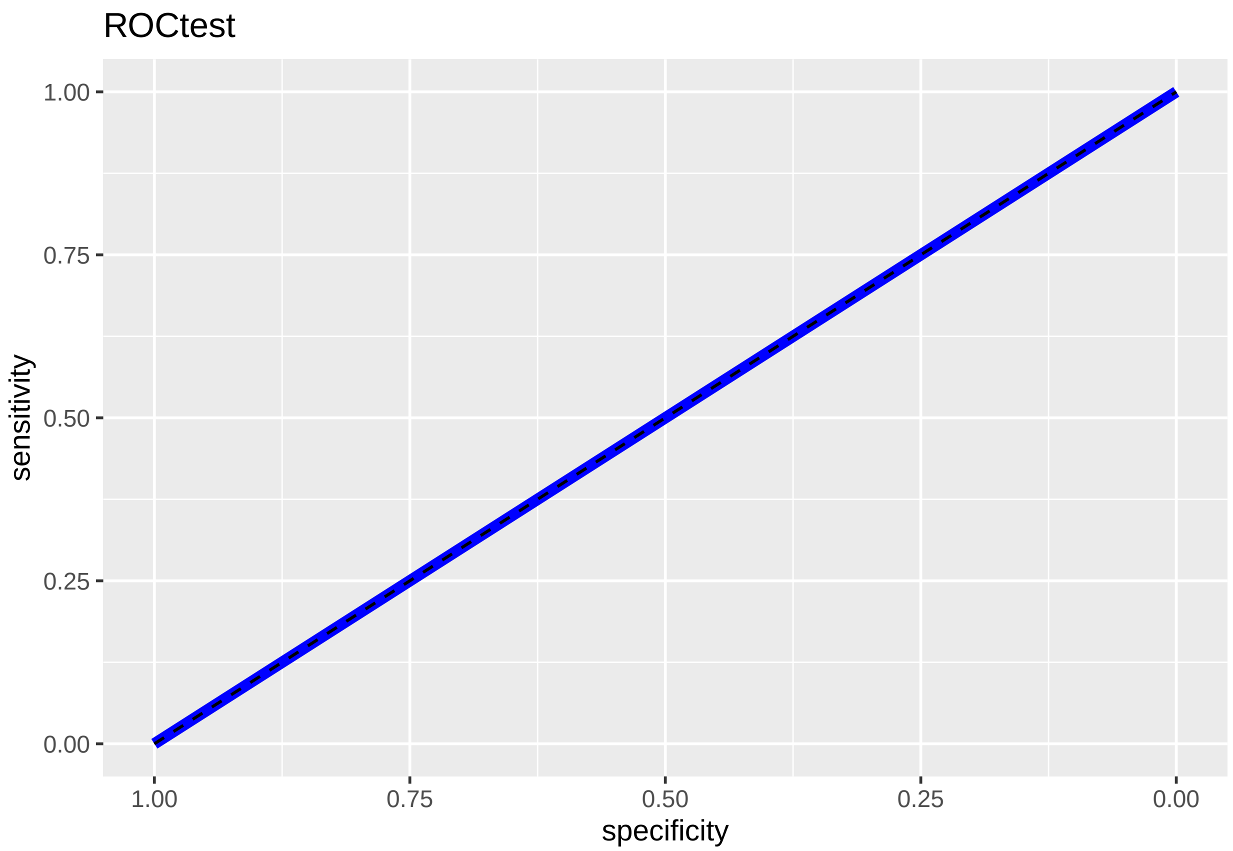
```
AUCknn <- AUCknn + AUC(knntest, datatest$ytest)
MSEknn <- MSEknn + MSE(as.numeric(knntest)-1, datatest$ytest)

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

##Logistic Regression
model.glm <- glm(ytrain~pctTOVOpponentMisc + pctDRBOpponentMisc +
                   paceTeamMisc + fg3mPerGameTeam + trbPerGameOpponent,
                 data = datatrain, family = binomial)
glmtest <- predict(model.glm, datatest, type = "response")
for (i in 1:length(glmtest)) {
  if(glmtest[i] <= 0.5){
    glmtest[i] <- as.factor(0)
  }
  if(glmtest[i] > 0.5){
    glmtest[i] <- as.factor(1)
  }
}
ConfusMatglm[[4]] <- ConfusionMatrix(factor(glmtest, levels=min(datatest$ytest):max(datatest$ytest)), fa
ConfusMatglm[[4]]
```

```
##       y_pred
## y_true   0   1
##      0   0   8
##      1   0 204
```

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

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

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



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

```
##Discriminant Models
##Linear Discriminant
model.lda <- lda(ytrain~ 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
## 0.04416404 0.95583596
##
## Group means:
##      winsTeam nrtgTeamMisc marginVictoryTeam     Ranking drtgTeamMisc
## 0  1.52311083   1.54272899        1.53214483 -1.40683346 -1.240434433
## 1 -0.04590246  -0.03311871       -0.03287429  0.03437097  0.002918694
##   pctFG2PerGameTeam pctFGPerGameTeam pctEFGTeamOppMisc pctFGPerGameOpponent
## 0         1.1802171       1.05460469        -1.1040720         -1.054422172
## 1        -0.0484582      -0.04385391        -0.0120674         -0.008730272
##   pctEFGTeamMisc ortgTeamMisc blkPerGameOpponent pctFG2PerGameOpponent
## 0     1.00545385   1.14689493        -1.23331701           -0.99470820
## 1    -0.05078135  -0.04683684         0.04676216           -0.02150005
##
## Coefficients of linear discriminants:
##                              LD1
## winsTeam              -0.4450422
## nrtgTeamMisc          -7.1267489
## marginVictoryTeam      7.2797369
## Ranking               -0.1918858
## drtgTeamMisc           0.4644367
## pctFG2PerGameTeam     -1.4815095
## pctFGPerGameTeam       0.1586297
## pctEFGTeamOppMisc      0.4863068
## pctFGPerGameOpponent  -0.2826037
## pctEFGTeamMisc         1.3315318
## ortgTeamMisc          -0.4199219
## blkPerGameOpponent     0.3670045
## pctFG2PerGameOpponent -0.2974499
```

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

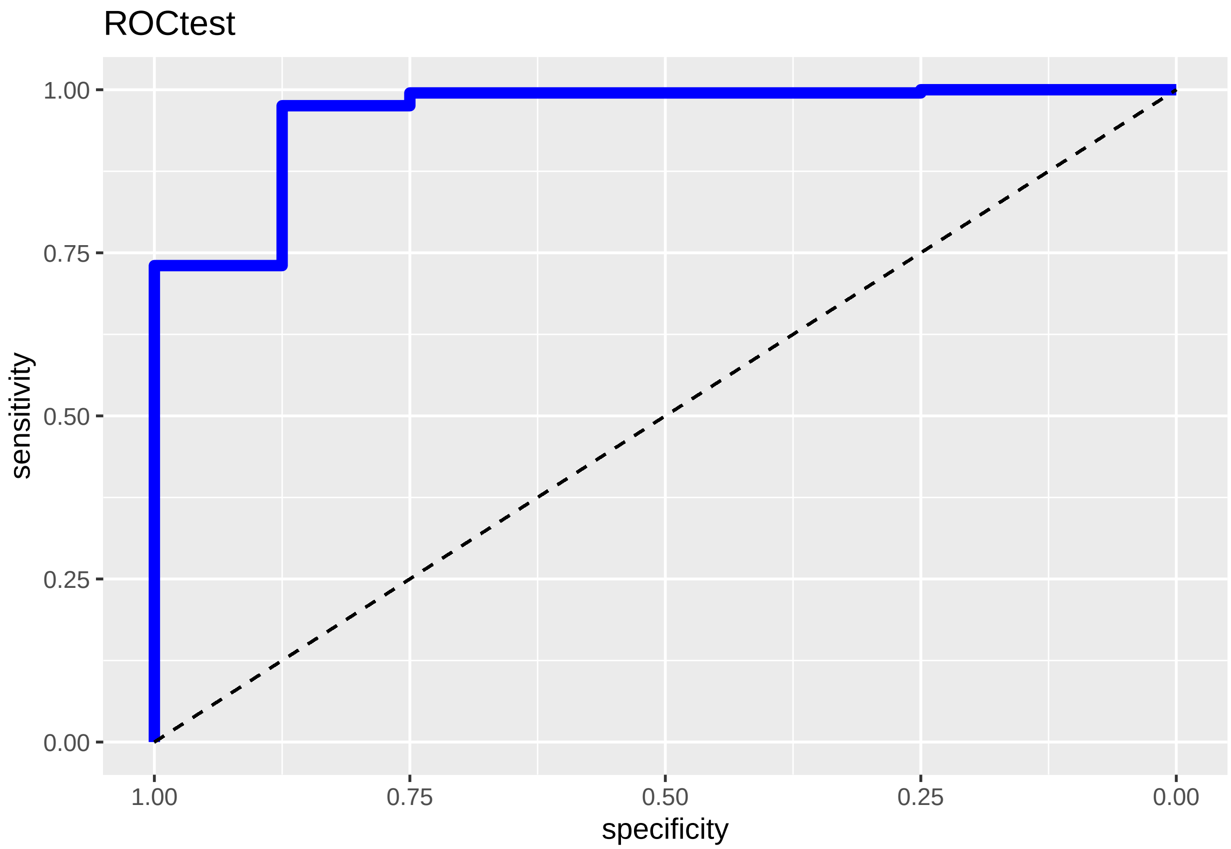
```
##       y_pred
## y_true   0   1
##      0   2   6
##      1   0 204
```

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

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

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

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



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

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

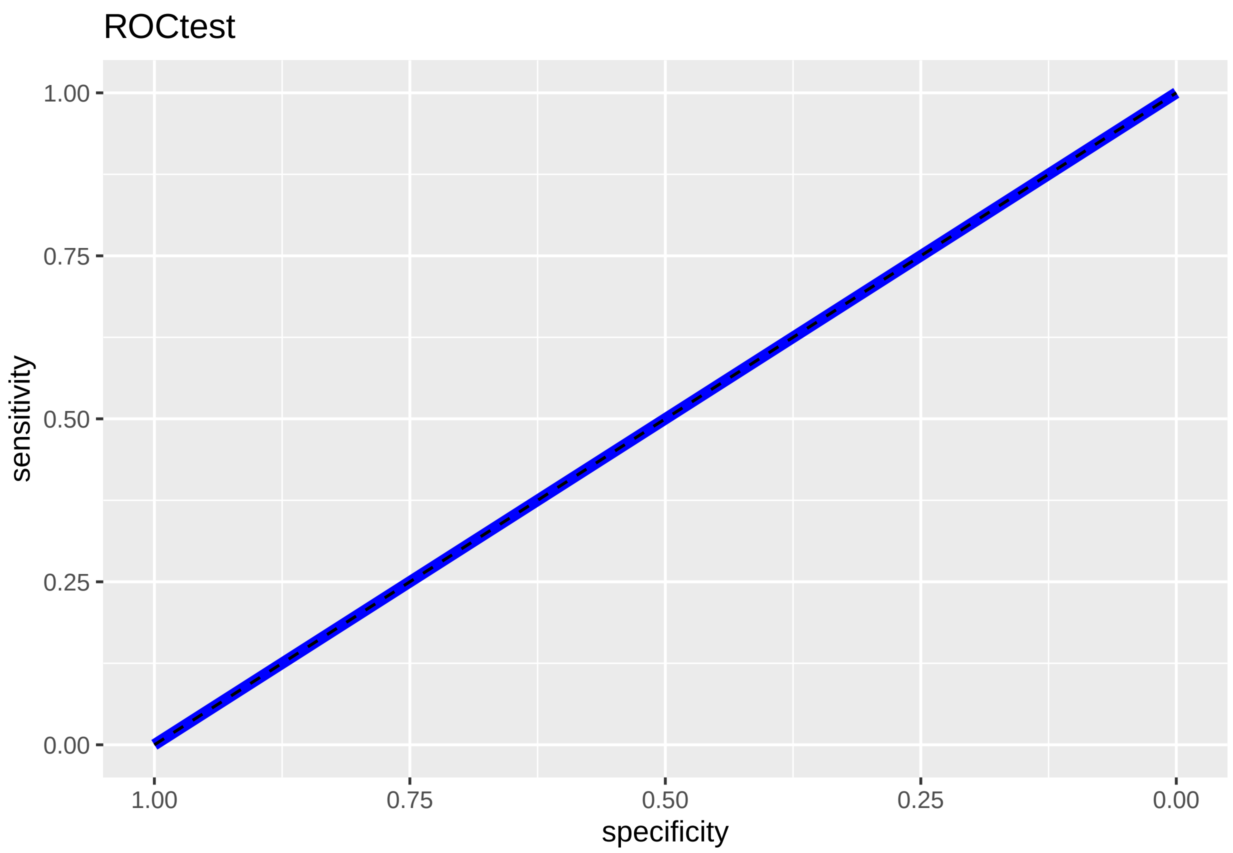
```
##         y_pred
## y_true   0   1
##      0   0   8
##      1   0 204
```

```
Accuracyknn <- Accuracyknn + ifelse(is.nan(Accuracy(knntest, datatest$ytest)),0,Accuracy(knntest, datate
Precisionknn <- Precisionknn + ifelse(is.nan(Precision(datatest$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)

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

```
## [1] 0.04254225
```

```
Accuracyglm/4
```

```
## [1] 0.9574577
```

```
Precisionglm/4
```

```
## [1] 0
```

```
Recallglm/4
```

```
## [1] 0
```

```
F1glm/4
```

```
## [1] 0
```

```
AUCglm/4
```

```
## [1] 0.5
```

```
ConfusMatglm
```

```
## [[1]]
##         y_pred
## y_true    0   1
##       0   0   4
##       1   0 207
##
## [[2]]
##         y_pred
## y_true    0   1
##       0   0  12
##       1   0 199
##
## [[3]]
##         y_pred
## y_true    0   1
##       0   0  12
##       1   0 200
##
## [[4]]
##         y_pred
## y_true    0   1
##       0   0   8
##       1   0 204
```

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

```
## [1] 1.100459
```

```
Accuracylda/4
```

```
## [1] 0.963354
```

```
Precisionlda/4
```

```
## [1] 0.75
```

```
Recallllda/4
```

```
## [1] 0.1666667
```

```
F1lda/4
```

```
## [1] 0.2714286
```

```
AUClda/4
```

```
## [1] 0.5821256
```

```
ConfusMatlda
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0   0   4
##      1   2 205
##
## [[2]]
##       y_pred
## y_true   0   1
##      0   2  10
##      1   0 199
##
## [[3]]
##       y_pred
## y_true   0   1
##      0   3   9
##      1   0 200
##
## [[4]]
##       y_pred
## y_true   0   1
##      0   2   6
##      1   0 204
```

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

```
## [1] 0.04018935
```

```
Accuracyknn/4
```

```
## [1] 0.9598107
```

```
Precisionknn/4
```

```
## [1] 0.4375
```

```
Recallknn/4
```

```
## [1] 0.2083333
```

```
F1knn/4
```

```
## [1] 0.2669526
```

```
AUCknn/4
```

```
## [1] 0.6004981
```

```
ConfusMatknn
```

```
## [[1]]
##       y_pred
## y_true   0   1
##      0   1   3
##      1   4 203
##
## [[2]]
##       y_pred
## y_true   0   1
##      0   3   9
##      1   1 198
##
## [[3]]
##       y_pred
## y_true   0   1
##      0   4   8
##      1   1 199
##
## [[4]]
##       y_pred
## y_true   0   1
##      0   0   8
##      1   0 204
```

```
##2020 Season Predictions
ytrain <- ceiling(MasterPerGame$finish/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  1  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"
```

```
##Logistic Regression Model suggests that the Los Angeles Lakers are a
##Championship Level team

##Discriminant Analysis
##Linear Discriminant
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
## 0.03427896 0.96572104
##
## Group means:
##      winsTeam nrtgTeamMisc marginVictoryTeam     Ranking drtgTeamMisc
## 0  1.52944855   1.55456417        1.54971702 -1.44159682  -1.27169931
## 1 -0.05428887  -0.05518037       -0.05500832  0.05117051   0.04513988
##   pctFG2PerGameTeam pctFGPerGameTeam pctEFGTeamOppMisc pctFGPerGameOpponent
## 0        1.27533168       1.22932275       -1.18968064          -1.14495335
## 1       -0.04526881      -0.04363569        0.04222857           0.04064094
##   pctEFGTeamMisc ortgTeamMisc blkPerGameOpponent pctFG2PerGameOpponent
## 0      1.2179358   1.13378153        -1.06401476            -1.0520341
## 1     -0.0432315  -0.04024439         0.03776797             0.0373427
##
## Coefficients of linear discriminants:
##                               LD1
## winsTeam              -0.16837662
## nrtgTeamMisc          -6.02342085
## marginVictoryTeam      4.93246138
## Ranking               -0.04939791
## drtgTeamMisc          -0.19230405
## pctFG2PerGameTeam     -0.64932983
## pctFGPerGameTeam      -0.15312347
## pctEFGTeamOppMisc      0.66207216
## pctFGPerGameOpponent  -0.25391566
## pctEFGTeamMisc         0.45852720
## ortgTeamMisc           0.56237486
## blkPerGameOpponent     0.23855278
## pctFG2PerGameOpponent -0.40919199
```

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

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

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

##Thus we can see that our LDA model cannot predict a champion for this year

##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 1 1 1 1 1 1 1 1 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]))
  }
}

##Likewise, we see that we cannot predict a champion for this year
```