Binyong Liang /bl9m

```r
library(tidyverse)
tb<-read_csv('HaitiPixels.csv')
#View(tb)
attach(tb)
classcate<-as.factor(Class)
contrasts(classcate)
```

```
##                   Rooftop Soil Various Non-Tarp Vegetation
## Blue Tarp               0    0                0          0
## Rooftop                 1    0                0          0
## Soil                    0    1                0          0
## Various Non-Tarp        0    0                1          0
## Vegetation              0    0                0          1
```

```r
summary(classcate)
```

```
##        Blue Tarp          Rooftop             Soil Various Non-Tarp
##             2022             9903            20566             4744
##       Vegetation
##            26006
```

```r
is.factor(classcate)
```

```
## [1] TRUE
```

```r
levels(classcate)
```

```
## [1] "Blue Tarp"        "Rooftop"          "Soil"
## [4] "Various Non-Tarp" "Vegetation"
```

```r
new.levels <- c("Blue Tarp", "aNBT","aNBT","aNBT","aNBT")
cate2<-factor(new.levels[classcate])
levels(cate2)
```

```
## [1] "aNBT"      "Blue Tarp"
```

```r
summary(cate2)
```

```
##      aNBT Blue Tarp
##     61219      2022
```

```r
tb["cate"]<-cate2
attach(tb)
summary(tb)
```

```
##     Class                Red          Green            Blue
##  Length:63241       Min.   : 48   Min.   : 48.0   Min.   : 44.0
##  Class :character   1st Qu.: 80   1st Qu.: 78.0   1st Qu.: 63.0
##  Mode  :character   Median :163   Median :148.0   Median :123.0
##                     Mean   :163   Mean   :153.7   Mean   :125.1
##                     3rd Qu.:255   3rd Qu.:226.0   3rd Qu.:181.0
##                     Max.   :255   Max.   :255.0   Max.   :255.0
##         cate
##  aNBT     :61219
##  Blue Tarp: 2022
##
##
##
##
```

```r
#data split 50/50, this is for ROC and AUC part
set.seed(1)
train <- sample(1:nrow(tb), nrow(tb)/2)
training <- tb[train,]
testing <- tb[-train,]
summary(training$cate)
```

```
##      aNBT Blue Tarp
##     30605      1015
```

```r
summary(testing$cate)
```

```
##      aNBT Blue Tarp
##     30614      1007
```

```r
#with 10-fold CV
n = nrow(tb)
set.seed(2020)
permutation = sample(n)
slice = n/10
```

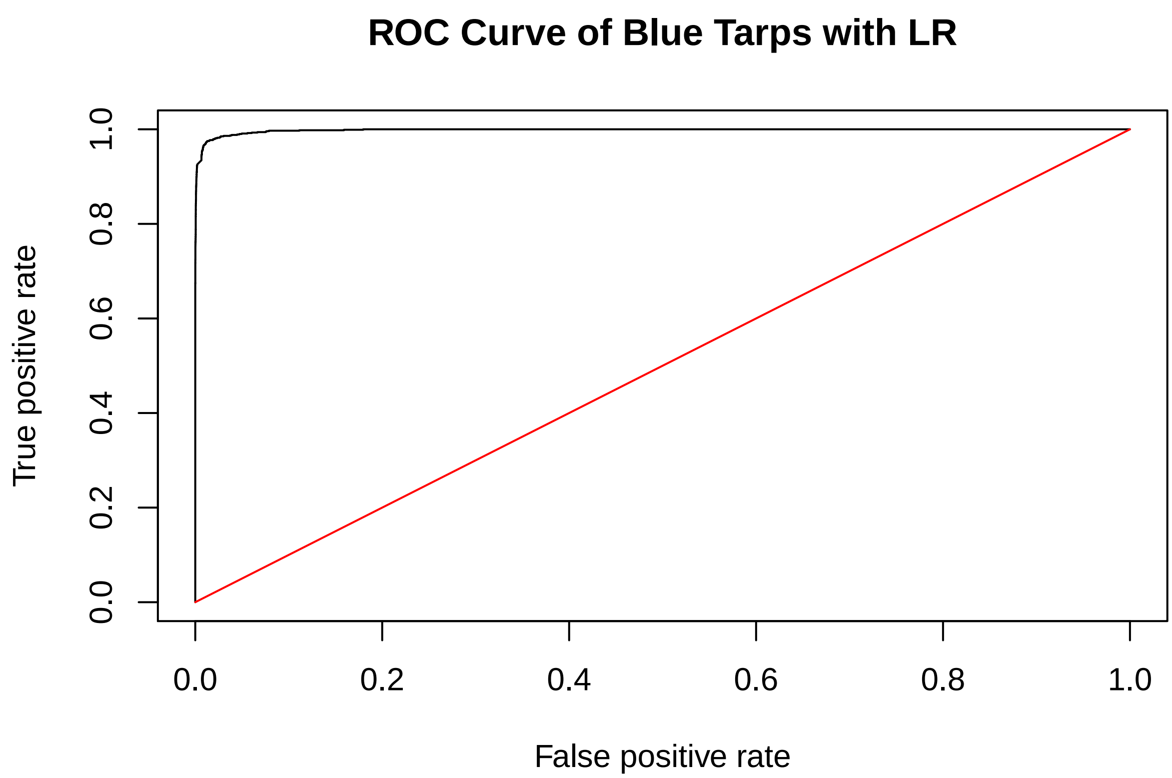
```r
#logistic regression

#with 10-fold CV
acc=0
for (i in 1:10) {
    test = permutation[((i-1)* slice +1) : (i*slice)]
    train = c(permutation[1:((i-1) * slice)], permutation[(i * slice + 1):n])
    glm.fit = glm(cate~Red+Green+Blue, data=tb, subset=train, family=binomial)
    glm.probs = predict(glm.fit, newdata=tb[test,], type ="response")
    glm.pred=rep("aNBT",nrow(tb[test,]))
    glm.pred[glm.probs>.5]="Blue Tarp"
    acc = acc + sum(glm.pred==tb[test,]$cate)/length(test)
}
acc = acc/10
acc
```

```
## [1] 0.9952562
```

```r
#ROC and AUC
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
glm.fits=glm(cate~Red+Green+Blue,data=training,family=binomial)
glm.probs=predict(glm.fits,newdata=testing,type="response")
pred <- prediction(glm.probs,testing$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with LR')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(pred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9981625
```

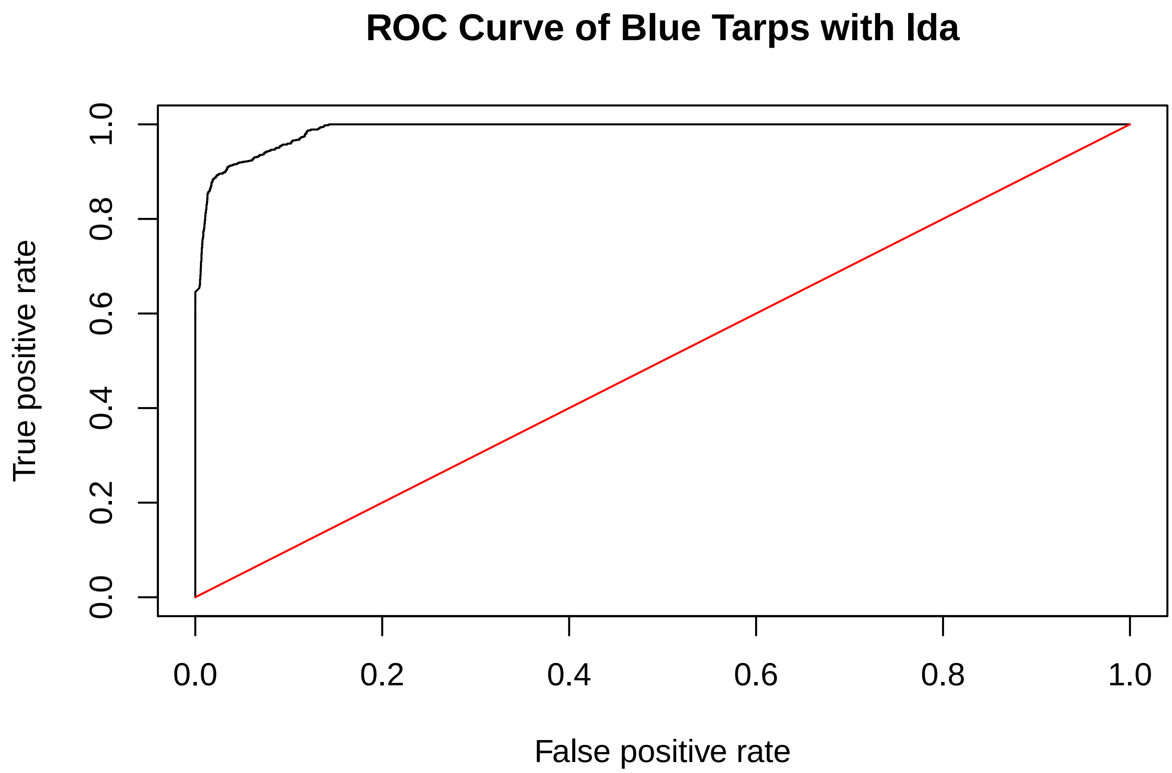
```r
#lda
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
#with 10-fold CV
acc=0
for (i in 1:10) {
    test = permutation[((i-1)* slice +1) : (i*slice)]
    train = c(permutation[1:((i-1) * slice)], permutation[(i * slice + 1):n])
    lda.fit = lda(cate~Red+Green+Blue, data=tb, subset=train)
    lda.pred=predict(lda.fit,tb[test,])
    acc = acc + sum(lda.pred$class==tb[test,]$cate)/length(test)
}
acc = acc/10
acc
```

```
## [1] 0.98395
```

```r
#ROC and AUC
lda.fit=lda(cate~Red+Green+Blue,data=training)
lda.pred=predict(lda.fit, testing)

pred <- prediction(lda.pred$posterior[,2],testing$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with lda')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```
auc<-performance(pred, measure = "auc")
auc@y.values
```

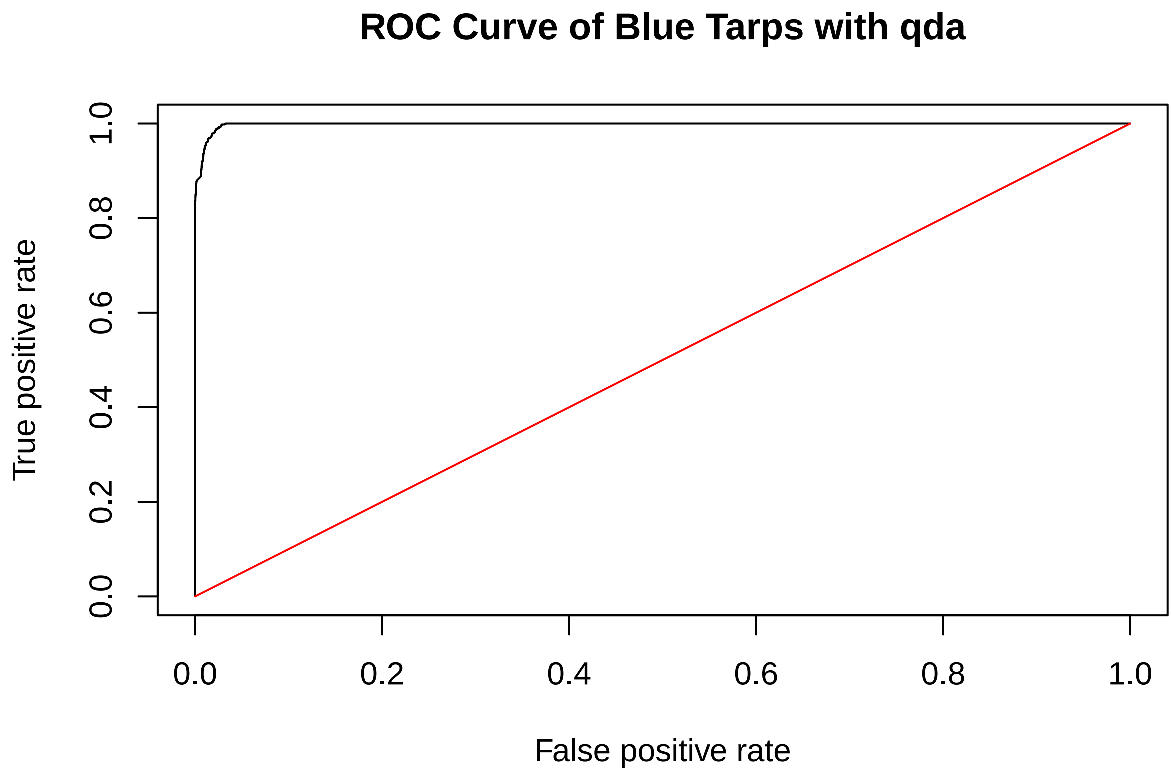
```
## [[1]]
## [1] 0.9888009
```

```
#qda

#10fold CV
acc=0
for (i in 1:10) {
    test = permutation[((i-1)* slice +1) : (i*slice)]
    train = c(permutation[1:((i-1) * slice)], permutation[(i * slice + 1):n])
    qda.fit = qda(cate~Red+Green+Blue, data=tb, subset=train)
    qda.pred=predict(qda.fit,tb[test,])
    acc = acc + sum(qda.pred$class==tb[test,]$cate)/length(test)
}
acc = acc/10
acc
```

```
## [1] 0.9945762
```

```
qda.fit=qda(cate~Red+Green+Blue,data=training)
qda.pred<-predict(qda.fit,testing)
```

```r
pred <- prediction(qda.pred$posterior[,2],testing$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with qda')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(pred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.998509
```

```r
#KNN
#10fold CV for K values from 1, 3, 5, ..., 25 (13 K-values total)
library(class)
acc<-numeric(13)
for (j in 0:12) {
  acc1<-0
  for (i in 1:10) {
    attach(tb)
    i=1
    test = permutation[((i-1)* slice +1) : (i*slice)]
    train = c(permutation[1:((i-1) * slice)], permutation[(i * slice + 1):n])
    train.X=cbind(Red,Green,Blue)[train,]
    test.X=cbind(Red,Green,Blue)[test,]
```

```r
    train.cate=cate[train]
    knn.pred=knn(train.X,test.X,train.cate,k=2*j+1)
    acc1 = acc1 + sum(knn.pred==cate[test])/length(test)
  }
  acc[j+1] = acc1/10
}
for (j in 0:12) {
  print(2*j+1)
  print(acc[j+1])
}
```

```
## [1] 1
## [1] 0.9955882
## [1] 3
## [1] 0.9972486
## [1] 5
## [1] 0.9969323
## [1] 7
## [1] 0.9969639
## [1] 9
## [1] 0.9966793
## [1] 11
## [1] 0.9969007
## [1] 13
## [1] 0.9965212
## [1] 15
## [1] 0.9966793
## [1] 17
## [1] 0.9961417
## [1] 19
## [1] 0.9960468
## [1] 21
## [1] 0.9960468
## [1] 23
## [1] 0.9957306
## [1] 25
## [1] 0.9957938
```

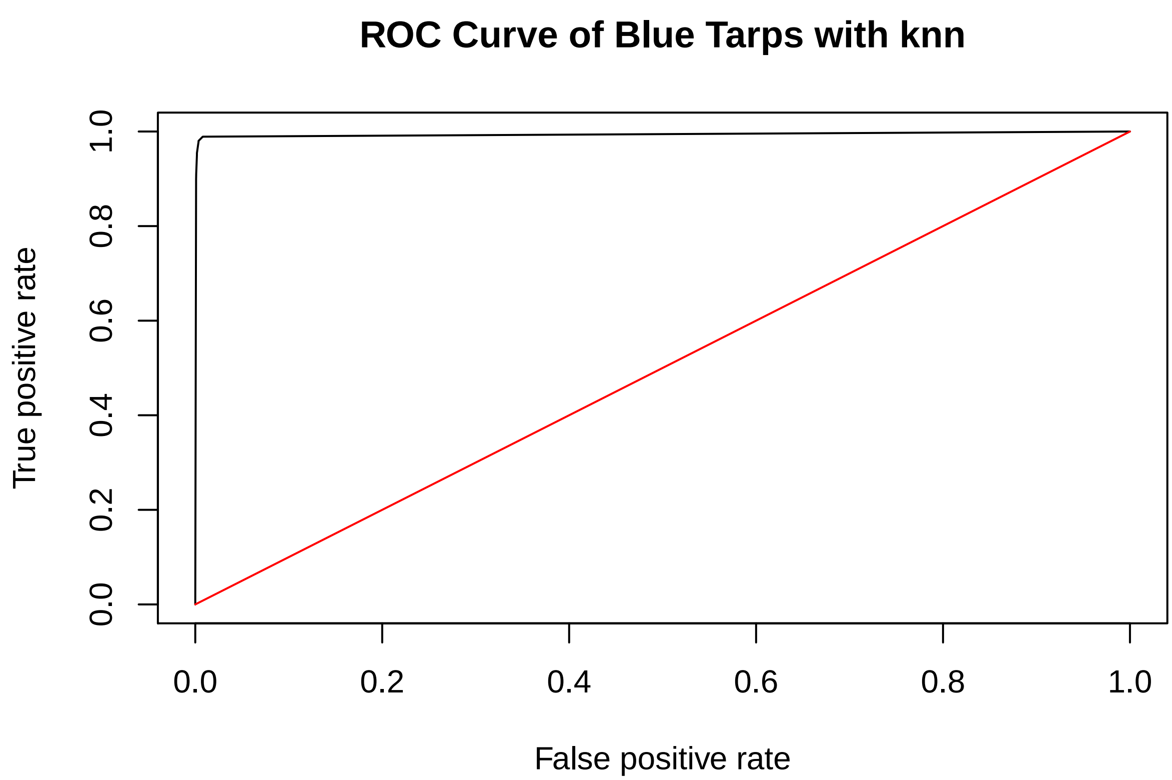
```r
#AUC and ROC

train.X=training[-c(1,5)]
test.X=testing[-c(1,5)]
train.cate=training$cate
knn.pred=knn(train.X,test.X,train.cate,k=3)
table(knn.pred, testing$cate)
```

```
##
## knn.pred     aNBT Blue Tarp
##   aNBT      30560        48
##   Blue Tarp    54       959
```

```r
mean(knn.pred==testing$cate)
```

```
## [1] 0.9967743
```

```r
knn.prob=knn(train.X,test.X,train.cate,k=3, prob=TRUE,use.all=TRUE)

knnprob = rep(0,nrow(testing))

for (i in 1:nrow(testing)) {
    if (knn.prob[i]=='aNBT') {
        knnprob[i] = attributes(knn.prob)$prob[i]
    } else {
        knnprob[i] = 1- attributes(knn.prob)$prob[i]
        }
}

pred <- prediction(1-knnprob,testing$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with knn')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(pred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9939038
```

```r
#Random Forest
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
#a tree size of 100 appears to be large enough.
acc<-numeric(100)
for (j in 1:100) {
  set.seed(1)
  rffit=randomForest(cate~Red+Green+Blue,data=tb,mtry=1,ntree=j)
  pred.rf = predict(rffit,newdata=tb)
  acc[j]=mean(pred.rf==tb$cate)
  }
acc
```

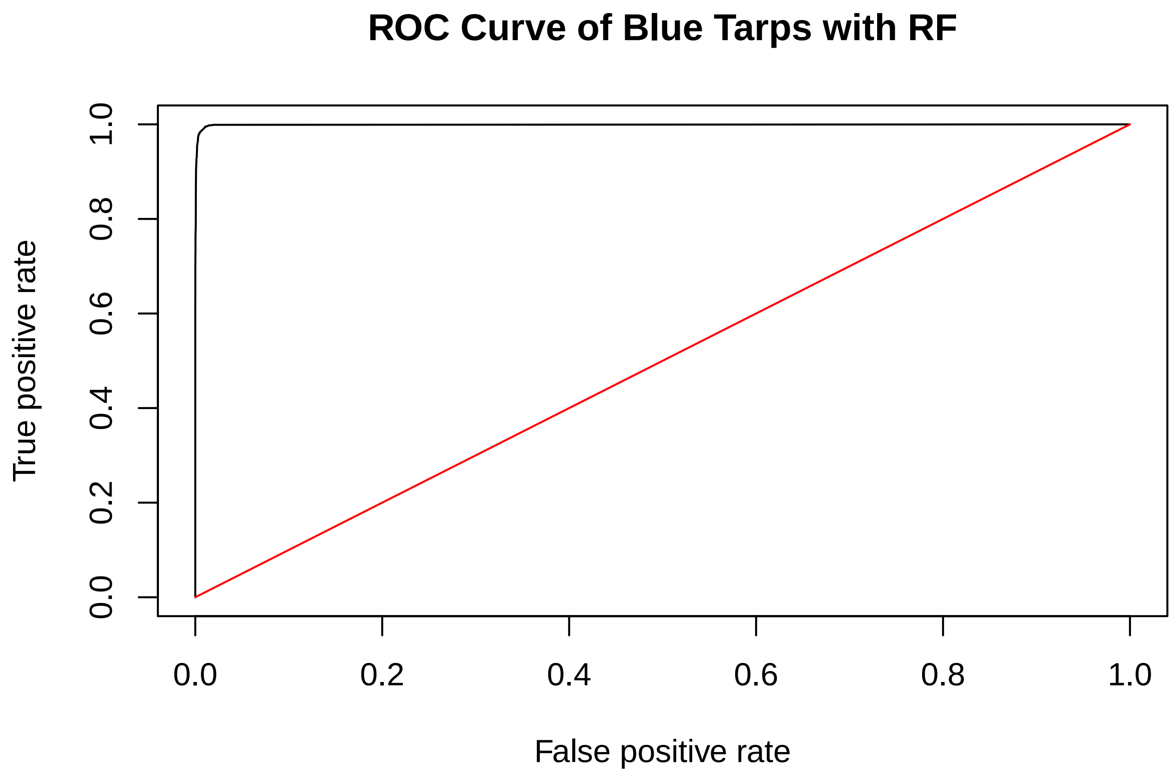
```
##   [1] 0.9971063 0.9958097 0.9984187 0.9972644 0.9986717 0.9985611 0.9990038
##   [8] 0.9987824 0.9990829 0.9990987 0.9991145 0.9991777 0.9991145 0.9991936
##  [15] 0.9991936 0.9992252 0.9992568 0.9992568 0.9993042 0.9992884 0.9992726
##  [22] 0.9992568 0.9993201 0.9993042 0.9992884 0.9993359 0.9993042 0.9993042
##  [29] 0.9992726 0.9993042 0.9992884 0.9993359 0.9992726 0.9992884 0.9992884
##  [36] 0.9992884 0.9993201 0.9993201 0.9993042 0.9993042 0.9993201 0.9993201
##  [43] 0.9993359 0.9993359 0.9993359 0.9993201 0.9993675 0.9993675 0.9993359
##  [50] 0.9993042 0.9993517 0.9993991 0.9993517 0.9993517 0.9993517 0.9993675
##  [57] 0.9993833 0.9993833 0.9993991 0.9993675 0.9993991 0.9994149 0.9994149
##  [64] 0.9994307 0.9993833 0.9994149 0.9993991 0.9993991 0.9994307 0.9994149
##  [71] 0.9994307 0.9994466 0.9994307 0.9994466 0.9994149 0.9994149 0.9993833
##  [78] 0.9993675 0.9993833 0.9993675 0.9993675 0.9993833 0.9993675 0.9993833
##  [85] 0.9993991 0.9994149 0.9993833 0.9994149 0.9993991 0.9994149 0.9993833
##  [92] 0.9993833 0.9993991 0.9993991 0.9993833 0.9993833 0.9994149 0.9993991
##  [99] 0.9993991 0.9994149
```

```r
plot(acc, col="white",xlab="Number of Trees")
lines(acc,col="black")
```


```
acc<-numeric(3)
for (j in 1:3) {
  acc1=0
  for (i in 1:10) {
    test = permutation[((i-1)* slice +1) : (i*slice)]
#    train = c(permutation[1:((i-1) * slice)], permutation[(i * slice + 1):n])
    rffit=randomForest(cate~Red+Green+Blue,data=tb[-test,],mtry=j,ntree=100)
    rf.pred=predict(rffit,newdata=tb[test,])
    acc1 = acc1 + mean(rf.pred==tb[test,]$cate)
  }
  acc[j] = acc1/10
}
acc
```

```
## [1] 0.9970114 0.9967109 0.9966793
```

```
#AUC
#data split 50/50, this is for ROC and AUC part for KNN
set.seed(1)
rffit=randomForest(cate~Red+Green+Blue,data=training,mtry=1,ntree=100)
rf.pred <- predict(rffit,testing,type="prob")
pred <- prediction(1-rf.pred[,1],testing$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with RF')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(pred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9990743
```

```r
#SVM
#linear kernel
library(e1071)
set.seed(1)
svmfit=svm(cate~Red+Green+Blue,data=tb, kernel="linear", cost=0.001)
svmfit
```

```
##
## Call:
## svm(formula = cate ~ Red + Green + Blue, data = tb, kernel = "linear",
##     cost = 0.001)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
##
## Number of Support Vectors:  4046
```

```r
summary(svmfit)
```

```
##
## Call:
## svm(formula = cate ~ Red + Green + Blue, data = tb, kernel = "linear",
##     cost = 0.001)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
##
## Number of Support Vectors:  4046
##
##  ( 2024 2022 )
##
##
## Number of Classes:  2
##
## Levels:
##  aNBT Blue Tarp
```

```r
tune.out=tune(svm,cate~Red+Green+Blue,data=tb,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.004712112
##
## - Detailed performance results:
##    cost       error   dispersion
## 1 1e-03 0.031719888 0.0021651098
## 2 1e-02 0.009187058 0.0013873004
## 3 1e-01 0.006214314 0.0011996999
## 4 1e+00 0.005012553 0.0009661910
## 5 5e+00 0.004712112 0.0009596871
## 6 1e+01 0.004727925 0.0009262483
```

```r
bestmod<-tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = cate ~ Red + Green + Blue,
```

```
##     data = tb, ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5,
##         10)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##
## Number of Support Vectors:  758
##
##  ( 380 378 )
##
##
## Number of Classes:  2
##
## Levels:
##  aNBT Blue Tarp
```

```r
#radial kernel
set.seed(1)
tuneradial=tune(svm, cate~Red+Green+Blue, data=tb, kernel="radial", ranges=list(cost=c(0.01,0.1,1,5,10)
summary(tuneradial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10     5
##
## - best performance: 0.002783011
##
## - Detailed performance results:
##     cost gamma       error   dispersion
## 1   0.01   0.1 0.019718190 0.0016087704
## 2   0.10   0.1 0.007004945 0.0011082431
## 3   1.00   0.1 0.004712127 0.0005906677
## 4   5.00   0.1 0.003984762 0.0005260796
## 5  10.00   0.1 0.003747576 0.0005274033
## 6   0.01   0.5 0.011052931 0.0012524907
## 7   0.10   0.5 0.005550180 0.0009172144
## 8   1.00   0.5 0.003921519 0.0005045528
## 9   5.00   0.5 0.003478761 0.0005110696
## 10 10.00   0.5 0.003399702 0.0004788134
## 11  0.01   1.0 0.008965689 0.0011255570
## 12  0.10   1.0 0.004886061 0.0008545642
## 13  1.00   1.0 0.003573638 0.0005437400
## 14  5.00   1.0 0.003257392 0.0005121845
## 15 10.00   1.0 0.003162515 0.0005427268
## 16  0.01   5.0 0.006720313 0.0011136920
## 17  0.10   5.0 0.003668519 0.0007142631
```

```
## 18  1.00   5.0 0.003099264 0.0005973106
## 19  5.00   5.0 0.002956949 0.0006194485
## 20 10.00   5.0 0.002783011 0.0004842725
```

```r
bestmod<-tuneradial$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = cate ~ Red + Green + Blue,
##     data = tb, ranges = list(cost = c(0.01, 0.1, 1, 5, 10), gamma = c(0.1,
##         0.5, 1, 5)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  467
##
##  ( 231 236 )
##
##
## Number of Classes:  2
##
## Levels:
##  aNBT Blue Tarp
```

```r
tuneradial$best.parameters
```

```
##    cost gamma
## 20   10     5
```

```r
set.seed(1)
tuneradial2=tune(svm, cate~Red+Green+Blue, data=tb, kernel="radial", ranges=list(cost=c(10,50,100),gamma
summary(tuneradial2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    50    10
##
## - best performance: 0.00256164
##
## - Detailed performance results:
##    cost gamma       error   dispersion
## 1   10     5 0.002783011 0.0004842725
```

```
## 2   50     5 0.002656511 0.0005716279
## 3  100     5 0.002593260 0.0005687000
## 4   10    10 0.002656511 0.0005860274
## 5   50    10 0.002561640 0.0005716365
## 6  100    10 0.002593265 0.0005538654
## 7   10    20 0.002593263 0.0004899806
## 8   50    20 0.002640704 0.0005480694
## 9  100    20 0.002688142 0.0004831429
```

```r
bestmod<-tuneradial2$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = cate ~ Red + Green + Blue,
##     data = tb, ranges = list(cost = c(10, 50, 100), gamma = c(5,
##         10, 20)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  50
##
## Number of Support Vectors:  423
##
##  ( 206 217 )
##
##
## Number of Classes:  2
##
## Levels:
##  aNBT Blue Tarp
```

```r
#best model: cost=50, gamma=10

#polynomial kernel
set.seed(1)
tunepoly=tune(svm,cate~Red+Green+Blue, data=tb, kernel="polynomial", ranges=list(cost=c(0.01,0.1,1,5,10
summary(tunepoly)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      1
##
## - best performance: 0.004759551
##
## - Detailed performance results:
```

```
##     cost degree       error   dispersion
## 1   0.01      1 0.013076939 0.0014200830
## 2   0.10      1 0.006736133 0.0012477524
## 3   1.00      1 0.005423685 0.0009251069
## 4   5.00      1 0.004870238 0.0009362370
## 5  10.00      1 0.004759551 0.0009019370
## 6   0.01      2 0.015148380 0.0014038194
## 7   0.10      2 0.009218679 0.0012427447
## 8   1.00      2 0.006688690 0.0012716442
## 9   5.00      2 0.006340813 0.0011843123
## 10 10.00      2 0.006277567 0.0011330277
## 11  0.01      3 0.017109117 0.0015084655
## 12  0.10      3 0.014816312 0.0014432729
## 13  1.00      3 0.008696872 0.0011925146
## 14  5.00      3 0.006214319 0.0012249409
## 15 10.00      3 0.005676694 0.0011117620
## 16  0.01      5 0.017915549 0.0019174633
## 17  0.10      5 0.015148368 0.0016307066
## 18  1.00      5 0.011432435 0.0014510632
## 19  5.00      5 0.009519119 0.0013698406
## 20 10.00      5 0.008965677 0.0012604919
```

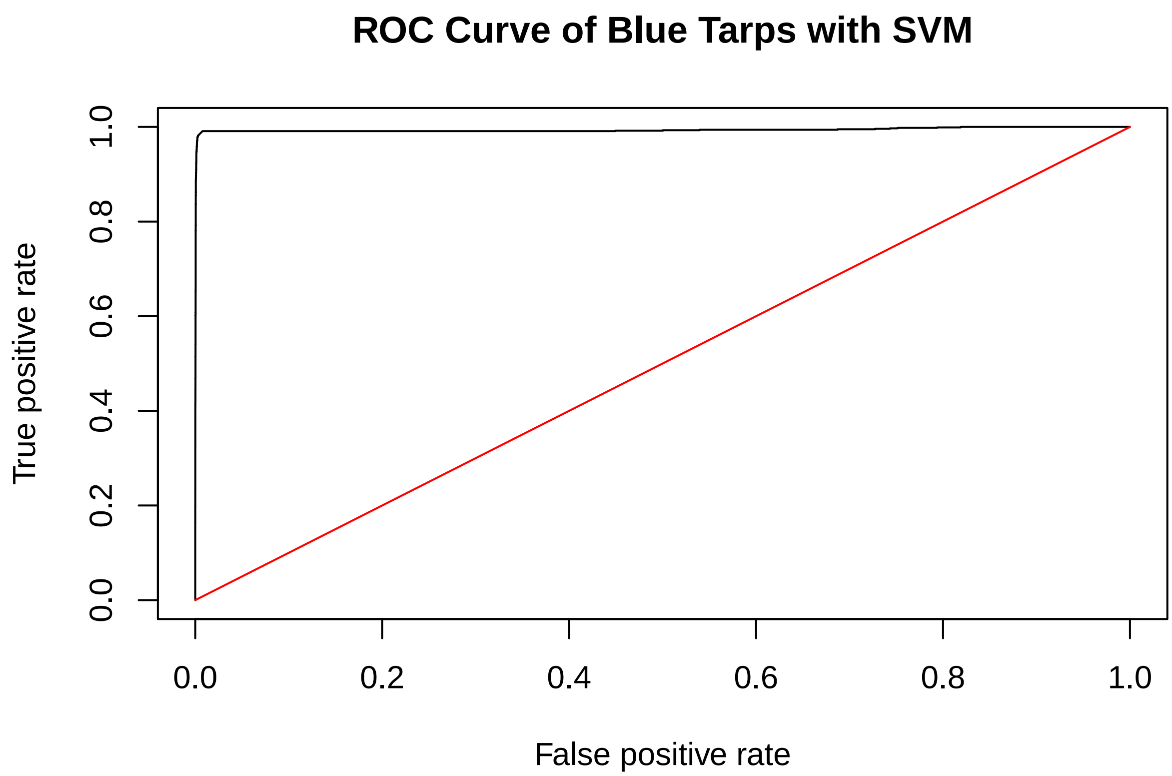
```r
bestmod<-tunepoly$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = cate ~ Red + Green + Blue,
##     data = tb, ranges = list(cost = c(0.01, 0.1, 1, 5, 10), degree = c(1,
##         2, 3, 5)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  1
##      coef.0:  0
##
## Number of Support Vectors:  781
##
##  ( 393 388 )
##
##
## Number of Classes:  2
##
## Levels:
##  aNBT Blue Tarp
```

```r
tunepoly$best.parameters
```

```
##   cost degree
## 5   10      1
```

```r
#best kernel: radial, with cost=50, gamma=10
#AUC
svmfit<-svm(cate~Red+Green+Blue, data=training, kernel="radial", cost=50, gamma=10,probability=TRUE)
svm.pred<-predict(svmfit,testing,probability=TRUE)
svmprob<-attr(svm.pred,"probabilities")
svmpred <- prediction(svmprob[,2],testing$cate)
roc_result <- performance(svmpred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with SVM')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(svmpred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9936511
```

The next section is for hold-out data. I will use the already determined the best model to refit to the training data set to get the best parameter sets. These models will be fit to the hold-out data.

```r
#KNN is K=3
#knn.pred=knn(train.X,test.X,train.cate,k=3)
#knn.prob=knn(train.X,test.X,train.cate,k=3, prob=TRUE,use.all=TRUE)
#LDA:
lda.fit=lda(cate~Red+Green+Blue,data=tb)
```

```
#QDA:
qda.fit=qda(cate~Red+Green+Blue,data=tb)
#Logistic regression:
lr.fit=glm(cate~Red+Green+Blue,data=tb,family=binomial)
#Random Forest:
set.seed(1)
rf.fit=randomForest(cate~Red+Green+Blue,data=tb,mtry=1,ntree=100)
#SVM:
set.seed(1)
svmfit<-svm(cate~Red+Green+Blue, data=tb, kernel="radial", cost=50, gamma=10,probability=TRUE)
```

Data clean up and combining

```
df<-read_delim("Hold+Out+Data/orthovnir067_ROI_Blue_Tarps.txt",skip=7,delim=" ")
df<-df[,-c(1:7,11:13)]
colnames(df)<-c("Red","Green","Blue")
cate <- rep("Blue Tarp",nrow(df))
df["cate"]<-cate

df2<-read_delim("Hold+Out+Data/orthovnir069_ROI_Blue_Tarps.txt",skip=7,delim=" ")
df2<-df2[,-c(1:7,11:13)]
colnames(df2)<-c("Red","Green","Blue")
cate <- rep("Blue Tarp",nrow(df2))
df2["cate"]<-cate


df3<-read_delim("Hold+Out+Data/orthovnir078_ROI_Blue_Tarps.txt",skip=7,delim=" ")
df3<-df3[,-c(1:7,11:13)]
colnames(df3)<-c("Red","Green","Blue")
cate <- rep("Blue Tarp",nrow(df3))
df3["cate"]<-cate


df4<-read_delim("Hold+Out+Data/orthovnir057_ROI_NON_Blue_Tarps.txt",skip=7,delim=" ")
df4<-df4[,-c(1:7,11:13)]
colnames(df4)<-c("Red","Green","Blue")
cate <- rep("aNBT",nrow(df4))
df4["cate"]<-cate

df5<-read_delim("Hold+Out+Data/orthovnir067_ROI_NOT_Blue_Tarps.txt",skip=7,delim=" ")
df5<-df5[,-c(1:7,11:13)]
colnames(df5)<-c("Red","Green","Blue")
cate <- rep("aNBT",nrow(df5))
df5["cate"]<-cate

df6<-read_delim("Hold+Out+Data/orthovnir069_ROI_NOT_Blue_Tarps.txt",skip=7,delim=" ")
df6<-df6[,-c(1:7,11:13)]
colnames(df6)<-c("Red","Green","Blue")
cate <- rep("aNBT",nrow(df6))
df6["cate"]<-cate

df7<-read_delim("Hold+Out+Data/orthovnir078_ROI_NON_Blue_Tarps.txt",skip=7,delim=" ")
df7<-df7[,-c(1:7,11:13)]
```

```r
colnames(df7)<-c("Red","Green","Blue")
cate <- rep("aNBT",nrow(df7))
df7["cate"]<-cate

testdf<-rbind(df,df2,df3,df4,df5,df6,df7)
nrow(testdf)
```

```
## [1] 2004177
```

```r
ncol(testdf)
```

```
## [1] 4
```

testing on the hold-out data

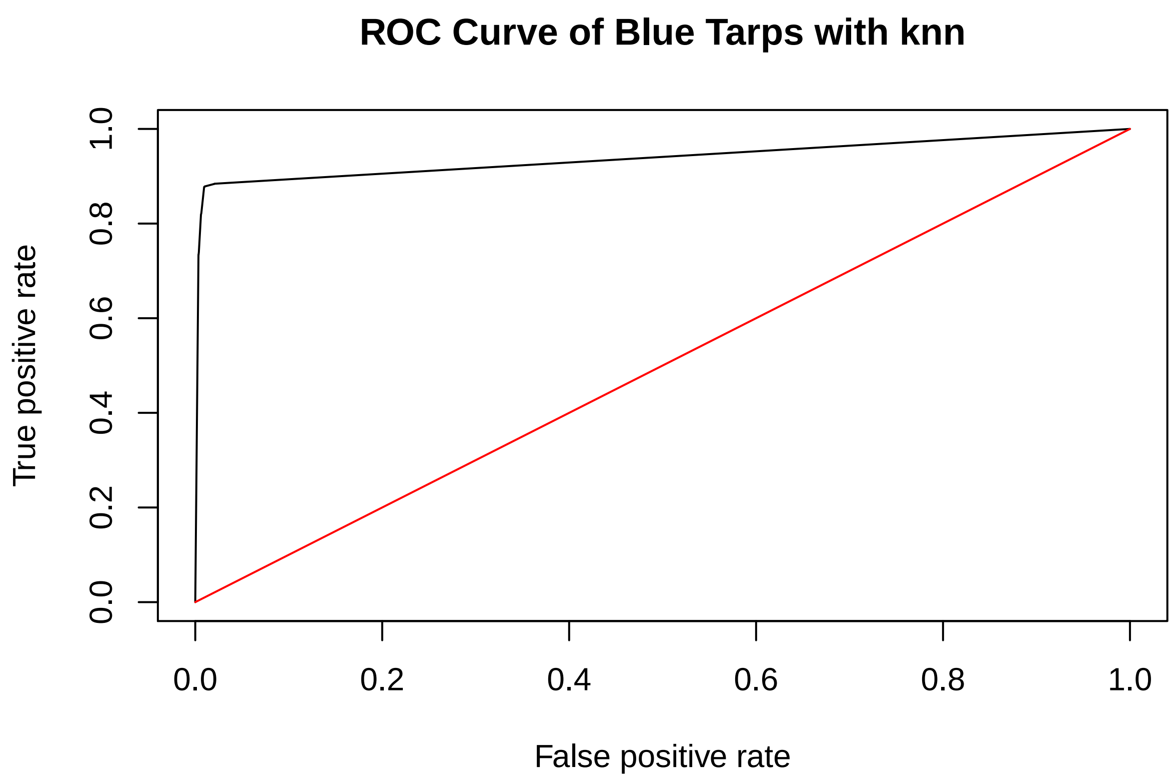
```r
#KNN at K=3
train_X <- tb[-c(1,5)]
test_X <- testdf[-4]
train_Y <- tb$cate

knn.prob=knn(train_X,test_X,train_Y,k=3, prob=TRUE,use.all=TRUE)
confs<-table(knn.prob,testdf$cate)
mean(knn.prob==testdf$cate)
```

```
## [1] 0.9924448
```

```r
(confs[1,1]+confs[2,2])/nrow(testdf)
```

```
## [1] 0.9924448
```

```r
knnprob = rep(0,nrow(testdf))
for (i in 1:nrow(testdf)) {
    if (knn.prob[i]=='aNBT') {
        knnprob[i] = attributes(knn.prob)$prob[i]
    } else {
        knnprob[i] = 1- attributes(knn.prob)$prob[i]
        }
}

pred <- prediction(1-knnprob,testdf$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with knn')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```
auc<-performance(pred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9386579
```

LDA

```
testdf$Red<-as.numeric(testdf$Red)
testdf$Green<-as.numeric(testdf$Green)
testdf$Blue<-as.numeric(testdf$Blue)

lda.pred=predict(lda.fit, testdf)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

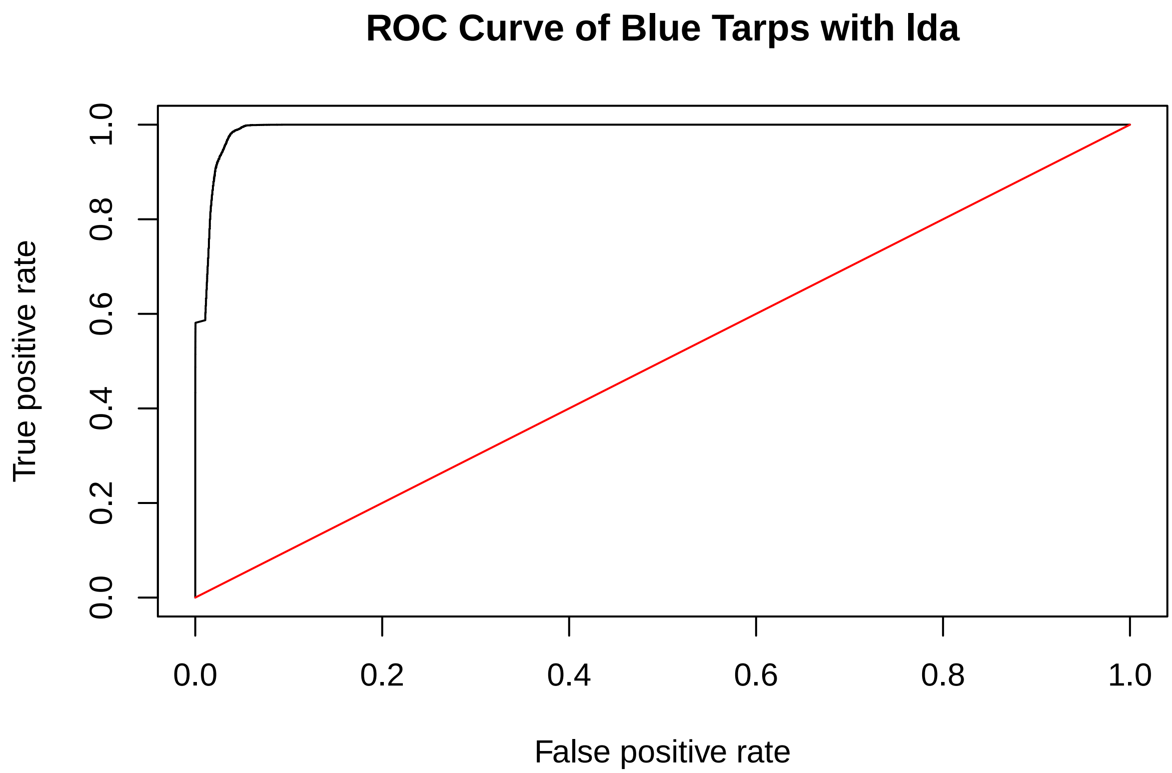
```
lda.class=lda.pred$class
confs<-table(lda.class,testdf$cate)
mean(lda.class==testdf$cate)
```

```
## [1] 0.9817496
```

```r
(confs[1,1]+confs[2,2])/nrow(testdf)
```

```
## [1] 0.9817496
```

```r
pred <- prediction(lda.pred$posterior[,2],testdf$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with lda')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(pred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9921155
```

QDA

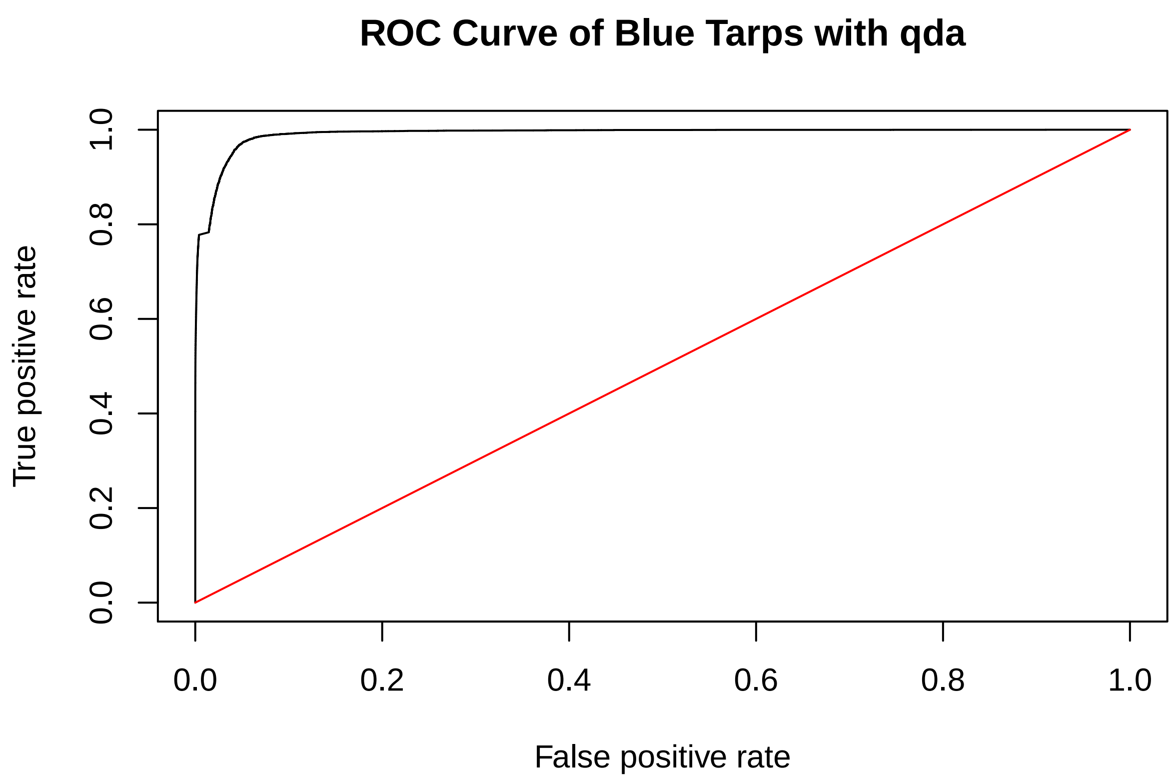
```r
qda.pred=predict(qda.fit, testdf)
qda.class=qda.pred$class
confs<-table(qda.class,testdf$cate)
mean(qda.class==testdf$cate)
```

```
## [1] 0.9959719
```

```
(confs[1,1]+confs[2,2])/nrow(testdf)
```

```
## [1] 0.9959719
```

```
pred <- prediction(qda.pred$posterior[,2],testdf$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with qda')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```
auc<-performance(pred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9915001
```

Logistic Regression

```
lr.fit=glm(cate~Red+Green+Blue,data=tb,family=binomial)
```

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

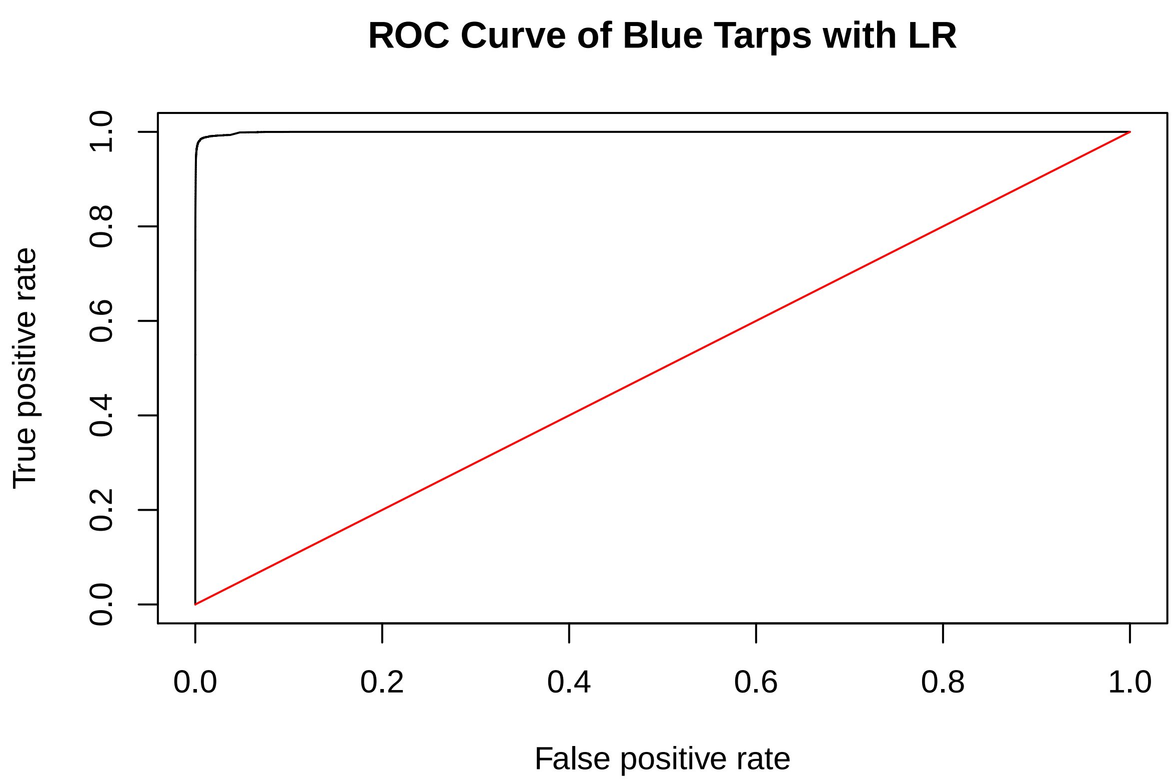
```r
lr.probs = predict(lr.fit, testdf, type ="response")
lr.pred=rep("aNBT",nrow(testdf))
lr.pred[lr.probs>.5]="Blue Tarp"
confs<-table(lr.pred,testdf$cate)
mean(lr.pred==testdf$cate)
```

```
## [1] 0.9897793
```

```r
(confs[1,1]+confs[2,2])/nrow(testdf)
```

```
## [1] 0.9897793
```

```r
pred <- prediction(lr.probs,testdf$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with LR')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(pred, measure = "auc")
auc@y.values
```

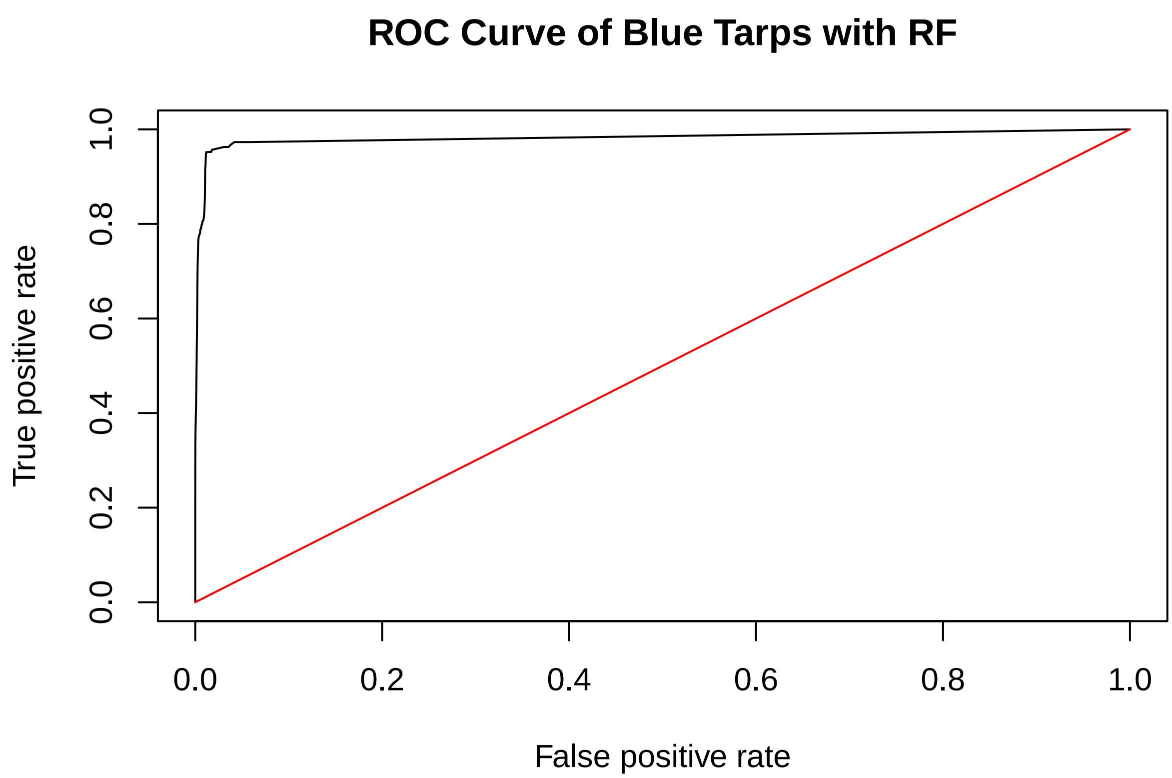
```
## [[1]]
## [1] 0.9994131
```

Random Forest

```
rf.pred=predict(rf.fit,newdata=testdf)
mean(rf.pred==testdf$cate)
```

```
## [1] 0.9946771
```

```
rf.pred <- predict(rf.fit,testdf,type="prob")
pred <- prediction(1-rf.pred[,1],testdf$cate)
roc_result <- performance(pred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with RF')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```
auc<-performance(pred, measure = "auc")
auc@y.values
```

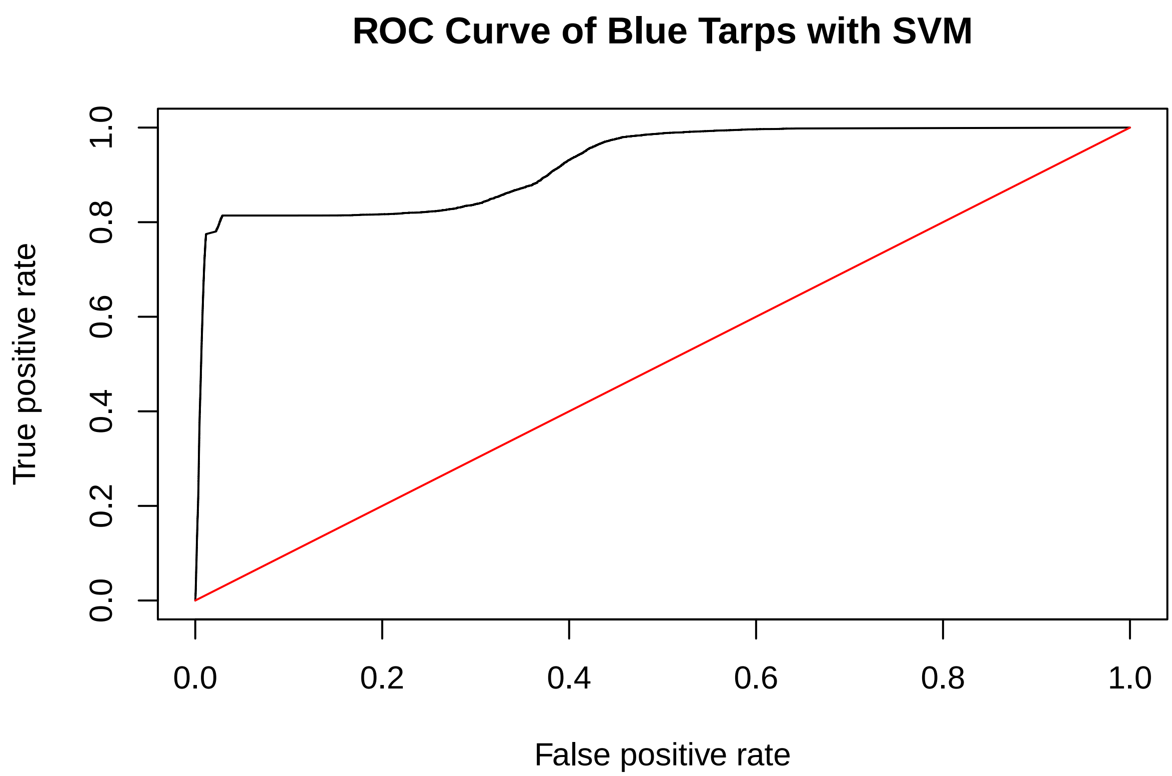
```
## [[1]]
## [1] 0.9826619
```

SVM

```
svm.pred<-predict(svmfit,testdf,probability=TRUE)
mean(svm.pred==testdf$cate)
```

```
## [1] 0.9904774
```

```r
svmprob<-attr(svm.pred,"probabilities")
svmpred <- prediction(svmprob[,2],testdf$cate)
roc_result <- performance(svmpred,'tpr','fpr')
plot(roc_result, main='ROC Curve of Blue Tarps with SVM')
lines(x= c(0,1), y= c(0,1), col = 'red')
```




```r
auc<-performance(svmpred, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.9238401
```