Project

Contents

Note: have not set seeds; have not split train and test datasets, have not finished data preprocessing

Project Description (est. 1 page, pt. 5)

Literature Review(est. 1 page, pt.10)

preprocessing and Summary statistics(est. 1 -2 pages, pt.10)

- check multicollinearity
- check outliers (`rs` and `pp`)
- check balance (`cn` and `mu`)

models that perform well on high-dimensional data

- SVM
- Random Forest
- Lasso (?)
- KNN regression

```
brca = read.csv("brca_data_w_subtypes.csv")
# head(brca)
```

```
dim(brca) # 705 rows, 1941 columns
```

```
## [1]  705 1941
```

```
names(brca)[1937:1941] # outcomes
```

```
## [1] "vital.status"      "PR.Status"         "ER.Status"
## [4] "HER2.Final.Status" "histological.type"
```

```r
# 1936 covariates: 860 copy number variations (cn), 249 somatic mutations (mu), 604 gene expressions (r
# order: rs 1:604, cn 605:1464, mu 1465:1713, pp 1714:1936

brca = brca[,-1937] # discard `vital.status`
names(brca)[1937:1940]
```

```
## [1] "PR.Status"         "ER.Status"         "HER2.Final.Status"
## [4] "histological.type"
```

```r
# unique values of responses
unique(brca$PR.Status)
```

```
## [1] "Positive"                    "Negative"
## [3] ""                            "Performed but Not Available"
## [5] "Indeterminate"               "Not Performed"
```

```r
unique(brca$ER.Status)
```

```
## [1] "Positive"                    "Negative"
## [3] ""                            "Performed but Not Available"
## [5] "Indeterminate"               "Not Performed"
```

```r
unique(brca$HER2.Final.Status)
```

```
## [1] "Negative"      ""              "Positive"      "Equivocal"
## [5] "Not Available"
```

```r
unique(brca$histological.type)
```

```
## [1] "infiltrating ductal carcinoma"  "infiltrating lobular carcinoma"
```

```r
# only use Negative and Positive for PR.status, ER.status, and HER2.Final.Status
# PR.status and ER.status are highly correlated
table(brca$PR.Status)
```

```
##
##                                       Indeterminate
##                         122                           4
##                    Negative               Not Performed
##                         193                          28
## Performed but Not Available                    Positive
##                           5                         353
```

```r
table(brca$ER.Status)
```

```
##
##                                       Indeterminate
##                         122                           2
##                    Negative               Not Performed
##                         135                          27
## Performed but Not Available                    Positive
##                           5                         414
```

```r
table(brca$HER2.Final.Status)
```

```
##
##                    Equivocal      Negative Not Available      Positive
##           145              9           457              8            86
```

```r
table(brca$histological.type)
```

```
##
##  infiltrating ductal carcinoma infiltrating lobular carcinoma
##                            574                            131
```

```r
# sub is the dataset we will use for modeling
# all of the null values are removed
sub = brca[(brca$PR.Status == "Positive" | brca$PR.Status == "Negative") &
           (brca$ER.Status == "Positive" | brca$ER.Status == "Negative") &
           (brca$HER2.Final.Status == "Positive" |
            brca$HER2.Final.Status == "Negative"),]
dim(sub)
```

```
## [1]  507 1940
```

```r
# the input variables have the indices below
# rs 1:604, cn 605:1464, mu 1465:1713, pp 1714:1936
```

check correlation

```r
rs = sub[1:604] # the subset that only contains rs
corr = round(cor(rs), 2) # correlation matrix
idx = data.frame(NA, NA, NA)
for (i in 1:nrow(corr)) {
  for (j in 1:nrow(corr)) {
    if (abs(corr[i, j]) > 0.8 & i < j) {
      idx[nrow(idx) + 1,] = c(i, j, corr[i, j])
    }
  }
}
idx = idx[-1,] # stores correlations that are greater than 0.8 -> multicollinearity
dim(idx)
```

```
## [1] 630   3
```

```r
names(idx) = c("i", "j", "corr")

# remove highly-correlated variables
rmv = idx[idx$j %in% names(sort(table(idx$j), decreasing = T)[1:61]),]$i
rmv = unique(rmv)
length(rmv)
```

```
## [1] 71
```

```r
rs = rs[,-rmv]
```

```r
cn = sub[605:1464]
corr = round(cor(cn), 2)
idx = data.frame(NA, NA, NA)
for (i in 1:nrow(corr)) {
  for (j in 1:nrow(corr)) {
    if (abs(corr[i, j]) > 0.8 & i < j) {
      idx[nrow(idx) + 1,] = c(i, j, corr[i, j])
    }
  }
}
idx = idx[-1,]
```

```
dim(idx)
```

```
## [1] 5054    3
```

```
names(idx) = c("i", "j", "corr")

rmv = idx[idx$j %in% names(sort(table(idx$j), decreasing = T)[1:683]),]$i
rmv = unique(rmv)
length(rmv)
```

```
## [1] 759
```

```
cn = cn[,-rmv]
```

```
mu = sub[1465:1713]
corr = round(cor(mu), 2)
idx = data.frame(NA, NA, NA)
for (i in 1:nrow(corr)) {
  for (j in 1:nrow(corr)) {
    if (abs(corr[i, j]) > 0.8 & i < j) {
      idx[nrow(idx) + 1,] = c(i, j, corr[i, j])
    }
  }
}
idx = idx[-1,]
dim(idx)
```

```
## [1] 0 3
```

```
names(idx) = c("i", "j", "corr")

rmv = idx[idx$j %in% names(sort(table(idx$j), decreasing = T)[1:10]),]$i
rmv = unique(rmv)
length(rmv)
```

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

```
# there is no multicollinearity within mu, so no variable is removed here
# mu = mu[,-rmv]
```

```
pp = sub[1714:1936]
corr = round(cor(pp), 2)
idx = data.frame(NA, NA, NA)
for (i in 1:nrow(corr)) {
  for (j in 1:nrow(corr)) {
    if (abs(corr[i, j]) > 0.8 & i < j) {
      idx[nrow(idx) + 1,] = c(i, j, corr[i, j])
    }
  }
}
idx = idx[-1,]
dim(idx)
```

```
## [1] 8 3
```

```
names(idx) = c("i", "j", "corr")

rmv = idx[idx$j %in% names(sort(table(idx$j), decreasing = T)[1:683]),]$i
```

```
rmv = unique(rmv)
length(rmv)
```

```
## [1] 8
```

```
pp = pp[,-rmv]
```

PR Status (Modeling, SVM and random Forest) (est. 2-3 pages, pt.20)

```
y = as.factor(sub$PR.Status)
y = ifelse(y == "Positive", 1, 0)
sub2 = cbind(rs, cn, mu, pp, y) # cleaned dataset with PR.status as response
dim(sub2)
```

```
## [1]  507 1099
```

```
set.seed(651978735)
n = dim(sub)[1]
test_size = as.integer(0.25 * n)
test_idx = sample(1:n, test_size) # 25% of the sample size

Xtest = sub2[test_idx, -ncol(sub2)]
Xtrain = sub2[-test_idx, -ncol(sub2)]

ytest = sub2[test_idx, ncol(sub2)]
ytrain = sub2[-test_idx, ncol(sub2)]
```

```
library(e1071)
svm.fit = svm(ytrain ~., data=Xtrain,
              type="C-classification", kernel="linear", scale=F, cost=1)
table("fitted" = svm.fit$fitted, "actual" = ytrain) # in-sample confusion matrix
```

```
##       actual
## fitted   0   1
##      0 133   0
##      1   0 248
```

```
pred = predict(svm.fit, newdata = Xtest)
table("fitted" = pred, "actual" = ytest)
```

```
##       actual
## fitted  0  1
##      0 34 14
##      1  9 69
```

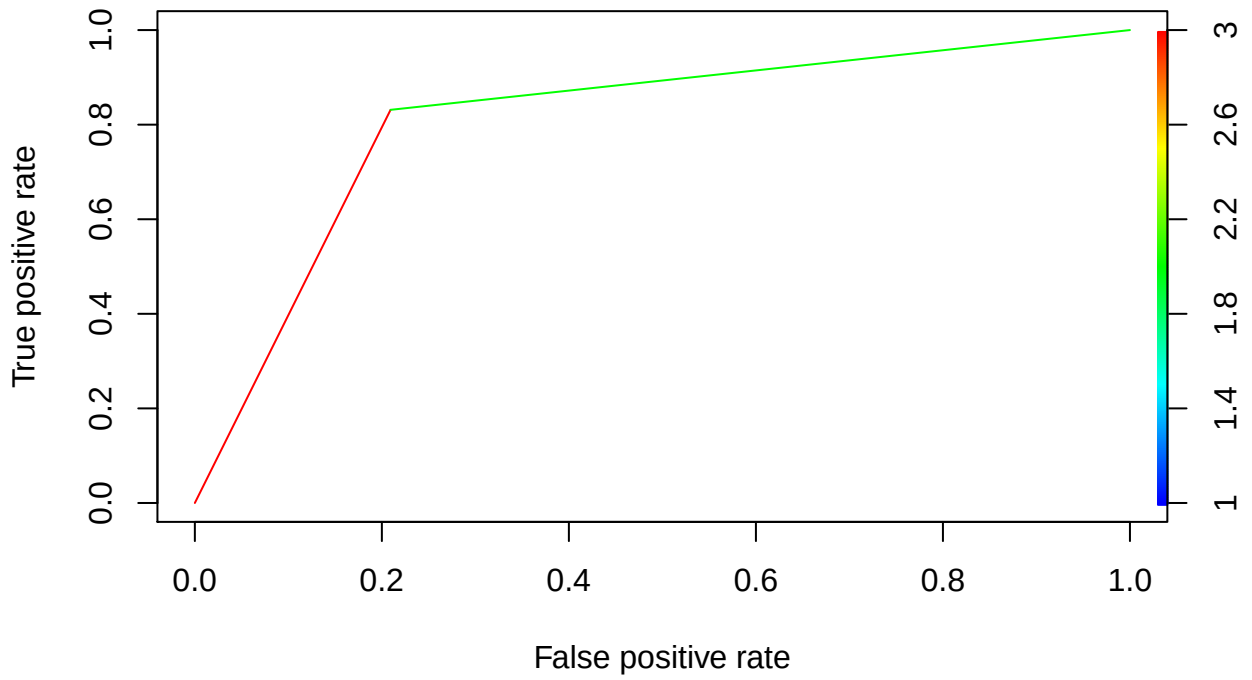
```
(34 + 69) / (34 + 69 + 14 + 9)
```

```
## [1] 0.8174603
```

```
library(ROCR)
roc = prediction(as.numeric(pred), ytest)
performance(roc, measure = "auc")@y.values[[1]]
```

```
## [1] 0.8110115
```

```
perf = performance(roc, "tpr", "fpr")
plot(perf, colorize = T)
```




```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
rf.fit = randomForest(Xtrain, as.factor(ytrain),
                      ntree=500,
                      mtry=10,
                      nodesize=10,
                      samplesize=400,
                      importance=TRUE)
```

```
pred = predict(rf.fit, Xtest)
table("fitted" = pred, "actual" = ytest)
```

```
##       actual
## fitted  0  1
##      0 33  2
##      1 10 81
```

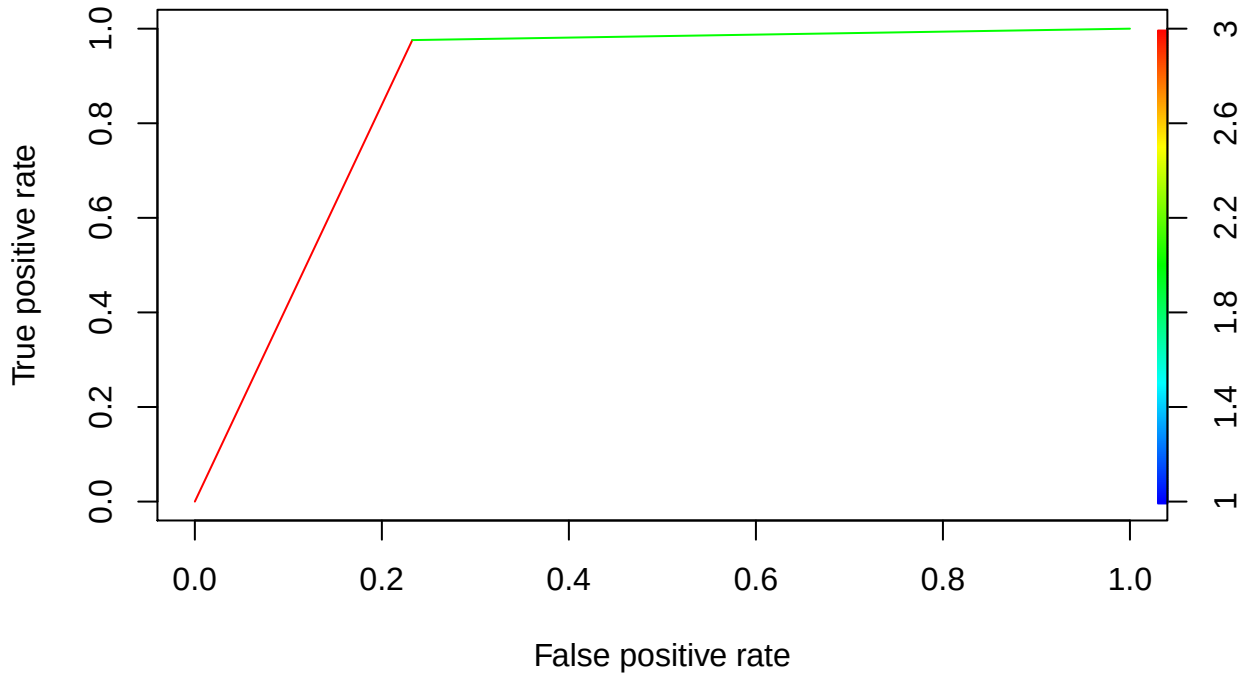
```
(33 + 81) / (33 + 81 + 2 + 10)
```

```
## [1] 0.9047619
```

```
roc = prediction(as.numeric(pred), ytest)
performance(roc, measure = "auc")@y.values[[1]]
```

```
## [1] 0.8716727
```

```
perf = performance(roc, "tpr", "fpr")
plot(perf, colorize = T)
```



see how SVM and Random Forest perform on the raw data `sub`

```
# test raw data
# test = randomForest(sub[1:1936], as.factor(sub2$y),
#                         ntree=500,
#                         mtry=10,
#                         nodesize=10,
#                         samplesize=400,
#                         importance=TRUE)
#
# pred = predict(test, sub[1:1936])
# table("fitted" = pred, "actual" = sub2$y)
# roc = prediction(as.numeric(pred), sub2$y)
# performance(roc, measure = "auc")@y.values[[1]]
#
# perf = performance(roc, "tpr", "fpr")
# plot(perf, colorize = T)
```

```
# test raw data
# test = svm(as.factor(y) ~., data=sub[1:1936],
#                 type="C-classification", kernel="linear", scale=F, cost=1)
# table("fitted" = test$fitted, "actual" = y)
# roc = prediction(as.numeric(svm.fit$fitted), y)
# performance(roc, measure = "auc")@y.values[[1]]
#
# perf = performance(roc, "tpr", "fpr")
# plot(perf, colorize = T)
```

Histological Type (to be decided) (est 2-3 pages, pt.20)

```
y = as.factor(sub$histological.type)
y = ifelse(y == "infiltrating lobular carcinoma", 1, 0)
```

```r
sub3 = cbind(rs, cn, mu, pp, y) # cleaned dataset with PR.status as response
dim(sub3)
```

```
## [1]  507 1099
```

```r
set.seed(651978735)
n = dim(sub)[1]
test_size = as.integer(0.25 * n)
test_idx = sample(1:n, test_size) # 25% of the sample size

Xtest = sub3[test_idx, -ncol(sub3)]
Xtrain = sub3[-test_idx, -ncol(sub3)]

ytest = sub3[test_idx, ncol(sub3)]
ytrain = sub3[-test_idx, ncol(sub3)]
```

```r
svm.fit = svm(ytrain ~., data=Xtrain,
              type="C-classification", kernel="linear", scale=F, cost=1)
table("fitted" = svm.fit$fitted, "actual" = ytrain)
```

```
##       actual
## fitted   0   1
##      0 342   0
##      1   0  39
```

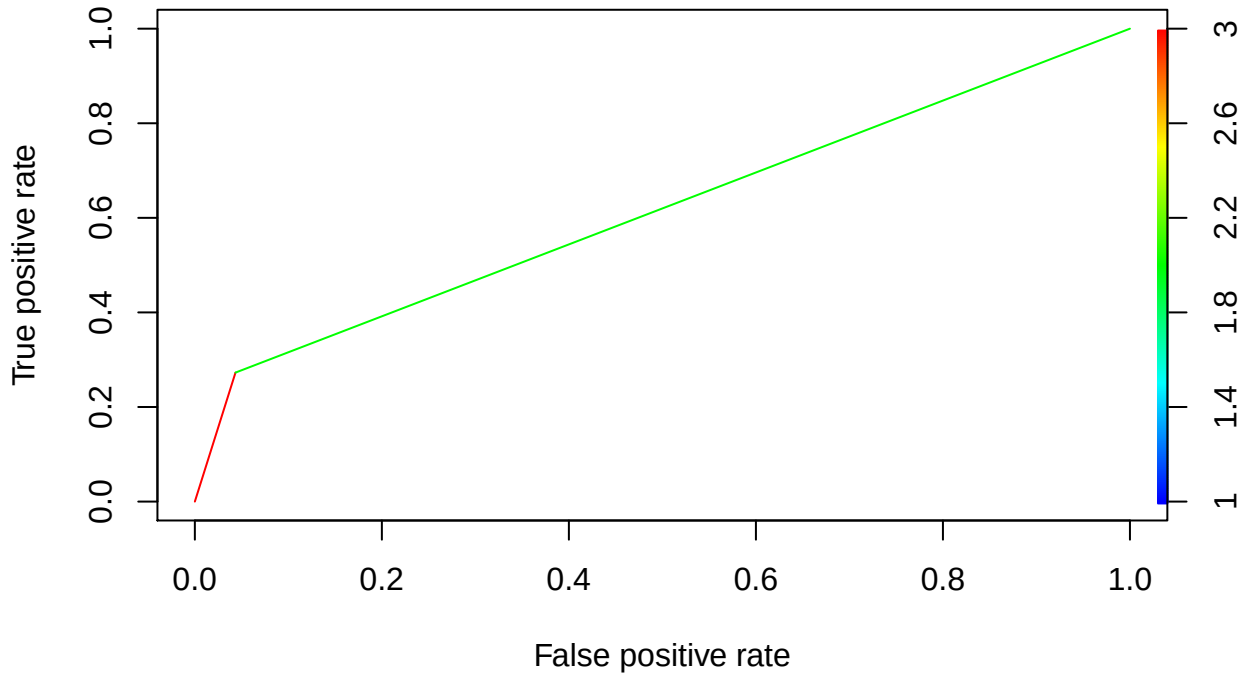
```r
pred = predict(svm.fit, newdata = Xtest)
table("fitted" = pred, "actual" = ytest)
```

```
##       actual
## fitted   0   1
##      0 110   8
##      1   5   3
```

```r
# library(ROCR)
roc = prediction(as.numeric(pred), ytest)
performance(roc, measure = "auc")@y.values[[1]]
```

```
## [1] 0.6146245
```

```r
perf = performance(roc, "tpr", "fpr")
plot(perf, colorize = T)
```



Variable Selection for All Outcomes (random forest?) (est. 2-3 pages. pt.20)

Using Random Forest, select the most important 50 variables, and make predictions based on these variables.

```
impt = importance(rf.fit)[order(importance(rf.fit)[,3], decreasing=TRUE),][1:50,]
vars = rownames(impt)
```

```
# sub4 is the cleaned dataset with all four response variables
sub4 = subset(sub, select = vars)
sub4 = cbind(sub4, sub[1937:1940])
sub4$PR.Status = as.factor(sub4$PR.Status)
sub4$histological.type = as.factor(sub4$histological.type)
sub4$ER.Status = as.factor(sub4$ER.Status)
sub4$HER2.Final.Status = as.factor(sub4$HER2.Final.Status)
```

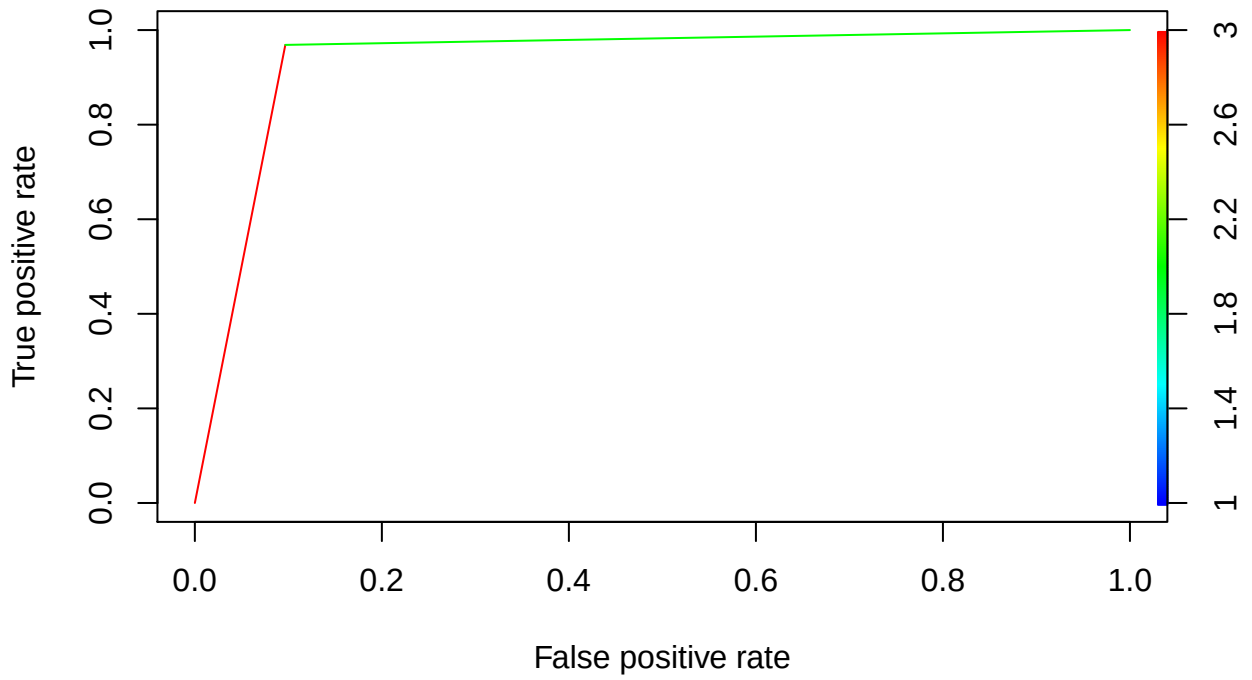
```
X = sub4[1:50] # input variables
svm.fit = svm(sub4$ER.Status ~., data=X,
              type="C-classification", kernel="linear", scale=F, cost=1)
table("fitted" = svm.fit$fitted, "actual" = y)
```

```
##           actual
## fitted       0   1
##   Negative 121   3
##   Positive 336  47
```

```
# library(ROCR)
roc = prediction(as.numeric(svm.fit$fitted), sub4$ER.Status)
performance(roc, measure = "auc")@y.values[[1]]
```

```
## [1] 0.9359471
```

```
perf = performance(roc, "tpr", "fpr")
plot(perf, colorize = T)
```



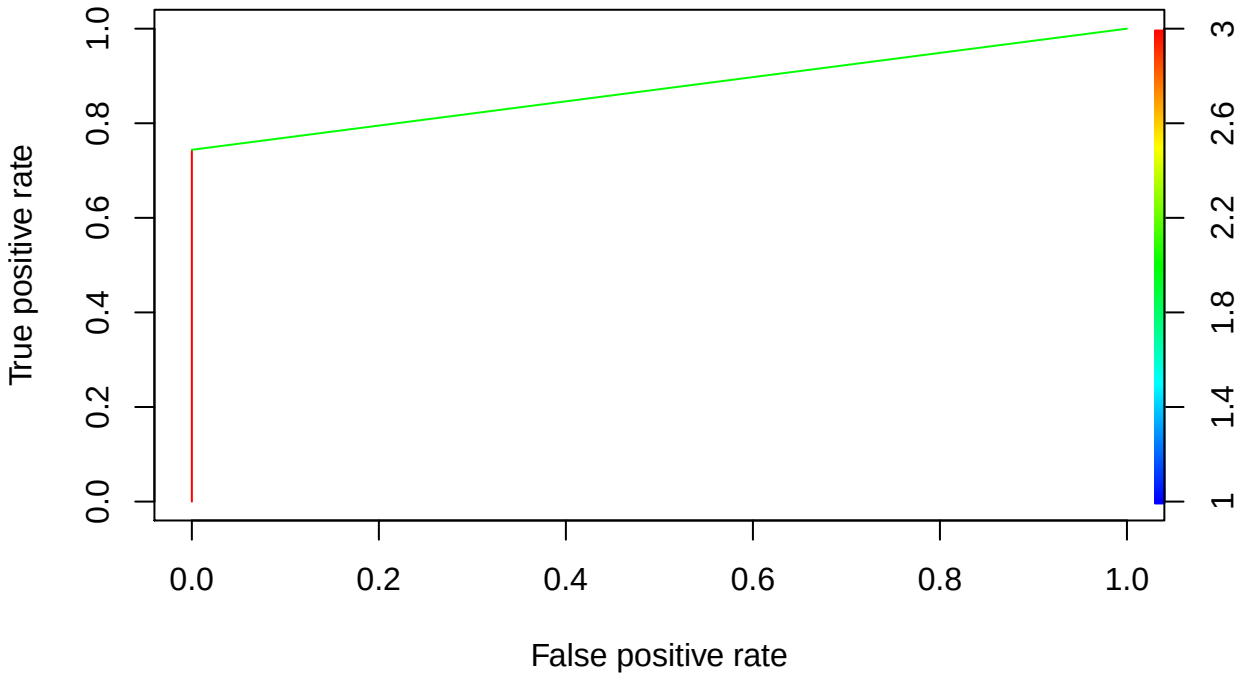

```
rf.fit = randomForest(X, sub4$HER2.Final.Status,
                      ntree=500,
                      mtry=7,
                      nodesize=10,
                      samplesize=400,
                      importance=TRUE)

pred = predict(rf.fit, X)
table("fitted" = pred, "actual" = sub4$HER2.Final.Status)
```

```
##           actual
## fitted     Negative Positive
##   Negative      425       21
##   Positive        0       61
```

```
roc = prediction(as.numeric(pred), sub4$HER2.Final.Status)
performance(roc, measure = "auc")@y.values[[1]]
```

```
## [1] 0.8719512
```

```
perf = performance(roc, "tpr", "fpr")
plot(perf, colorize = T)
```


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
1
1.4
1.8
2.2
2.6
3