STAT 432 Final Project

Contents

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

Literature Review

The title of our first relevant paper is Tumor characteristics and patient outcomes are similar between invasive lobular and mixed invasive ductal/lobular breast cancers but differ from pure invasive ductal breast cancers. A total of 4,336 individuals with IDC, ILC, and mixed breast tumors were detected between 1996 and 2006.

The Kaplan-Meier method was used extensively in this paper, and survival curves were constructed using it.Chi-square tests and Fisher's exact tests were used to compare clinical variables. The correlations between patient and tumor variables were summarized using contingency tables and investigated using Fisher's exact test as among three histologic groups. Patients with ILC and mixed breast cancers were more probable as IDC patients to have tumors that were estrogen receptor and progesterone receptor positive ($P < 0.001$ and $P< 0.05$, correspondingly).

After having read, we can conclude the following from the paper: first, despite being identified at lower clinical stages of infection, patients with IDC had the poorest long-term survival; second, individuals with ILC and "mixed" malignancies had a better prognosis than patients with IDC, despite having more advanced cancer. We were also motivated to utilize the log-rank test to estimate P values if necessary.

The second research article is Infiltrating lobular carcinoma of the breast: tumor characteristics and clinical outcome. We summarize that these patients do not have improved clinical outcomes as IDC patients when

ignoring the fact that ILC has a positive biologic pattern. Consequently, management decisions should be made based on the patient's and tumor's biologic characteristics, that instead of lobular histology.

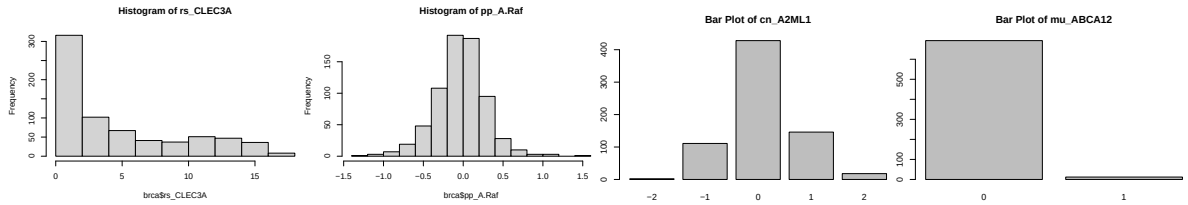
About statistical methods, the clinical and biologic features of lobular and ductal carcinoma were compared by contingency tables, Chi-square tests and Fisher's exact tests, which is similar with the method using in the first paper. To see if ILC was an independent predictive predictor for recurrence and death, researchers used multivariate analysis and Cox regression models. Tumor size, number of affected nodes, age, ER status, PgR status, DNA ploidy, S-phase, and histologic type were all considered in these analyses.

The findings of this huge dataset have shown that ILC and IDC are distinct entities with distinctive clinical histories and biologic features, yet there are no clinically important variations in survival. At the present, both kinds of breast cancer should be treated identically, and histologic subtype (lobular or ductal) should not be regarded a determinant in therapeutic decision-making or an essential prognostic or predictive factor at diagnosis. Emerging technologies such as high throughput genome mapping and microchip cDNA expression arrays may help to uncover molecular distinctions between these different types of breast cancer.

Summary Statistics and Data Preprocessing

Data Overview

The dataset has 705 observations and 1941 features (1936 predictors and 5 outcomes). There are four different kinds of predictors: `rs` (gene expression), `cn` (copy number variations), `mu` (mutations), and `pp` (protein levels). Among them, `rs` and `pp` are continuous variables, and `cn` and `mu` are categorical variables.



Remove Missing Values

According to the instruction, we dropped `vital.status`, and we only considered each response variable as a binary variable. Therefore, we treated the observations that had other outcomes as missing values and removed them from our dataset.

Then the dataset `sub` had 507 observations and 1940 features.

```
dim(sub)
```

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

Deal with Multicollinearity

One of the noticeable characteristics of the data is its high dimensionality. There are 1936 predictors, almost four times as many as there are observations. Therefore, it is essential to check correlation.

Since there are four kinds of predictors, it is unlikely that two variables coming from different kinds would be highly correlated. Also, to reduce the computational cost, we split the data into four subsets: `rs`, `cn`, `mu`, and `pp`, each of which contained only ond kind of predictors.

Then, we created the correlation matrix for each subset, and extracted variables that are highly-correlated with at least one other variable. Take `rs` as an example. The dataframe `idx` stores all matrix indices of highly-correlated variables and the corresponding correlation coefficients. If the i-th variable is highly-correlated with the j-th variable, then we only need one of them. Thus, we removed all variables with indices `i`. For `rs`, 94 predictors were removed. We applied the same process to the other three subsets. In total, 882 predictors were removed. There are 1059 predictors remained.

```
names(idx) = c("i", "j", "corr")
idx[1:3,]
```

```
##   i  j corr
## 2 3  4 0.94
## 3 5 56 0.84
## 4 9 10 0.95
```

```
# remove highly-correlated variables
rmv = unique(idx[,1])
length(rmv)
```

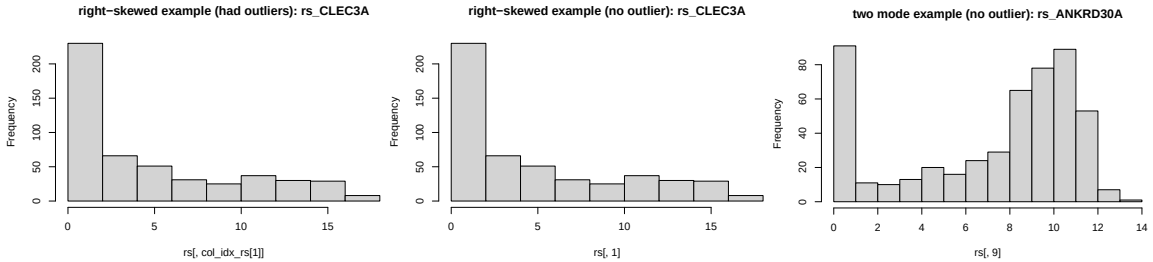
```
## [1] 94
```

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

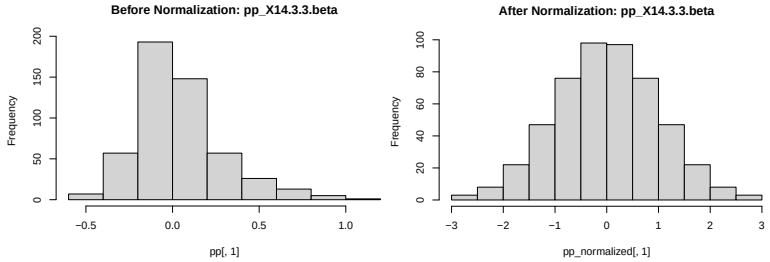
Continuous Variables

As mentioned before, `rs` and `pp` are continuous variables, so we should examine if there are any outliers. We first normalized the variable, and stored row and column indices if the data point was three standard deviations away from the mean. For the two subsets, `rs` had 100 outliers, and `pp` had no outlier.

We further looked into `rs` predictors that included outliers, and we found the vast majority of them had a long tail, mostly right and some left. In addition, a number of `rs` predictors that did not contain outliers also had a non-standard distribution. As a result, a log transformation of `rs` predictors would be beneficial.



Unlike `rs`, `pp` variables were distributed quite normally. However, many of the variables would contain outliers without normalization. Therefore, we normalized `pp` variables.



Categorical Variables

All four outcomes were more or less imbalanced, among which `histological.type` was the most imbalanced. Only 10% of the responses were ILC. In some cases, such imbalance would be problematic since models would learn nothing from the minority class. If our data also suffer from problems like this, we should resolve the imbalance by techniques like undersampling. However, if our models perform well enough on current data, no further action needs to be taken.

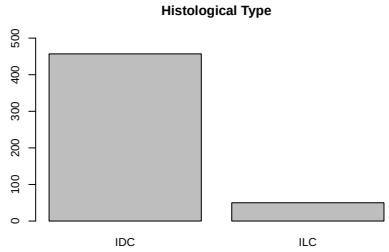
Fortunately, we trained our models first, and found the models obtained high enough accuracy and AUC scores. Therefore, we decided to not address imbalanced outcomes.

```
##
## Negative Positive
##  0.34714  0.65286
```

```
##
##  Negative  Positive
## 0.2445759 0.7554241
```

```
##
##  infiltrating ductal carcinoma infiltrating lobular carcinoma
##                     0.90138067                     0.09861933
```

```
##
##  Negative  Positive
## 0.8382643 0.1617357
```



Modeling `PR Status`

Before modeling, we split the train and test datasets. We used 25% of the samples (126) for testing and 75% of the samples (381) for training.

Support Vector Machine (SVM)

The goal of the project was to make classifications. Plus, we needed to alleviate "the curse of dimensionality". Therefore, we should choose classification models that perform well on high-dimensional data. Support vector machines are famous for its capability in high-dimensional spaces, so we first fitted a basic linear SVM, with the default `cost = 1` to see how it worked.

As the confusion matrix showed, the in-sample accuracy was 1.0, which implied that we might prefer the linear kernel to the radial kernel.

```
##           actual
## predicted  Negative Positive
##   Negative      133        0
##   Positive        0      248
```

Then we constructed two grids of tuning parameters for both linear and radial kernels, and we used 5-fold cross-validation to tune the parameters and to determine which kernel was better.

For the linear kernel, the best `C` was 0.001 with an in-sample accuracy of 0.8324.

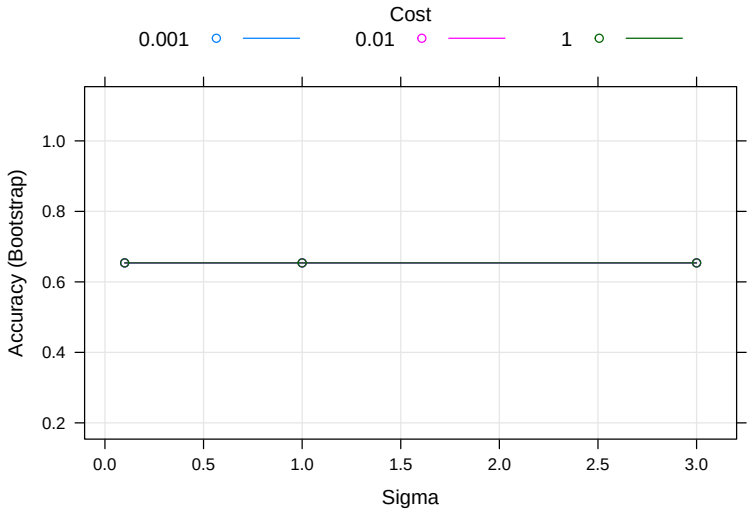
```
##       C
## 1 0.001
```

```
##       C  Accuracy     Kappa AccuracySD    KappaSD
## 1 0.001 0.8336029 0.6058770 0.02730027 0.06541415
## 2 0.010 0.7999388 0.5464954 0.02526173 0.06257166
## 3 0.050 0.8014261 0.5503720 0.02442031 0.06150767
## 4 0.100 0.8014261 0.5503720 0.02442031 0.06150767
## 5 0.500 0.8014261 0.5503720 0.02442031 0.06150767
```

```
## 6 1.000 0.8014261 0.5503720 0.02442031 0.06150767
```

For the radial kernel, the best `C` was 0.001 and the best `sigma` was 3. However, the figure showed that the radial SVM fitted poorly, since the accuracies remained the same and were merely 0.6677. The fact verified our hypothesis that a linear kernel would work better. Thus, we picked the linear SVM with `C` equals 0.001 to make classifications.

```
##   sigma     C
## 3     3 0.001
```



We made predictions for the test data, and printed the confusion table below. The accuracy was 0.9048. Thus, the linear SVM performed quite well.

```
##           actual
## predicted  Negative Positive
##   Negative       33        2
##   Positive       10       81
```

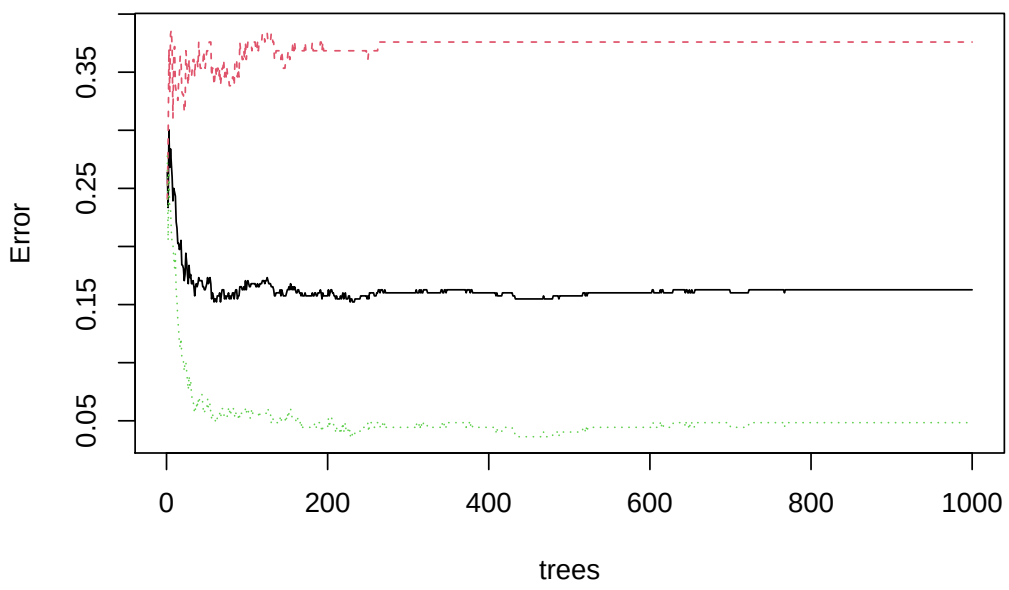
```
## [1] 0.9047619
```

Random Forest

Random forests are another classification model that is less vulnerable to "the curse of dimensionality". Since there were many parameters that needed to be tuned, we again utilized `caret` package to cross validate the model.

Before applying cross-validation, we should first determine `num.trees`, because it could not be included in the grid of parameters. Therefore, we first fitted a random forest using the `randomForest()` method with `ntree = 1000`, and plotted the error against trees.

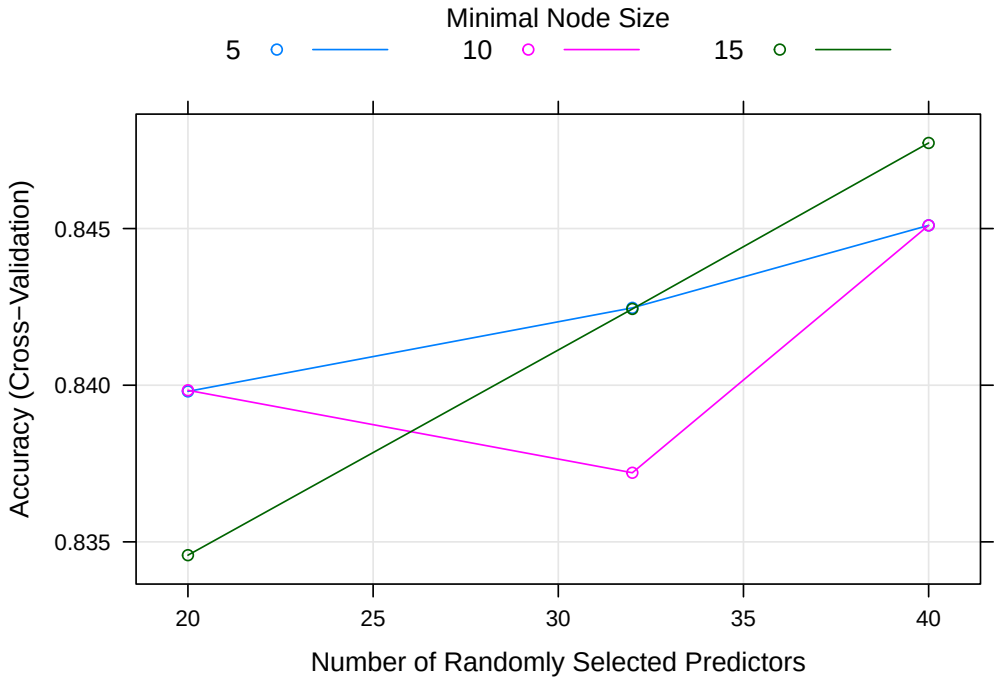
As the plot demonstrated, the error stopped decreasing after the number of trees reached around 500. Thus, `ntree = 500` should be sufficient for training.



In a 5-fold cross-validation, we tuned `mtry` and `min.node.size`. The output showed that the best parameters were `mtry = 40` and `min.node.size = 15` when `num.trees = 500` according to the test accuracy.

```
##   mtry splitrule min.node.size
## 9   40      gini            15
```

```
##   mtry splitrule min.node.size  Accuracy     Kappa AccuracySD    KappaSD
## 1   20      gini             5 0.8398027 0.6199996 0.01816714 0.04744967
## 2   20      gini            10 0.8398378 0.6191721 0.01181501 0.02631224
## 3   20      gini            15 0.8345746 0.6097325 0.01575637 0.03413845
## 4   32      gini             5 0.8424694 0.6272972 0.01375275 0.03444365
## 5   32      gini            10 0.8372062 0.6150392 0.01262152 0.02909492
## 6   32      gini            15 0.8424343 0.6268844 0.01948896 0.05058240
## 7   40      gini             5 0.8451009 0.6341821 0.01487847 0.03698468
## 8   40      gini            10 0.8451009 0.6356210 0.02177987 0.05139481
## 9   40      gini            15 0.8477325 0.6408985 0.01796807 0.04555969
```



The confusion matrix and the highest test accuracy, 0.9048, were shown here.

```
##           actual
## predicted  Negative Positive
##   Negative       33        2
##   Positive       10       81
```

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

Modeling `Histological Type`

To establish the Modeling for histological Type, We observe that the response variable `Histological Type` has only two levels "infiltrating lobular carcinoma" and "infiltrating ductal carcinoma". Thus, Logistic model become a good choice. To Peform a logistic Model, we first split the data into train and test data. Test data will be random selected 25% of the sample size.

Logistic Regression

Once, we attempted to fit logistic model with our data, the algorithm throw a warning said "algorithm did not converge". In other words, we are experience a perfect sepration. As the confusion matrix shows below, our model has 100% accuracy and AUC with 1.

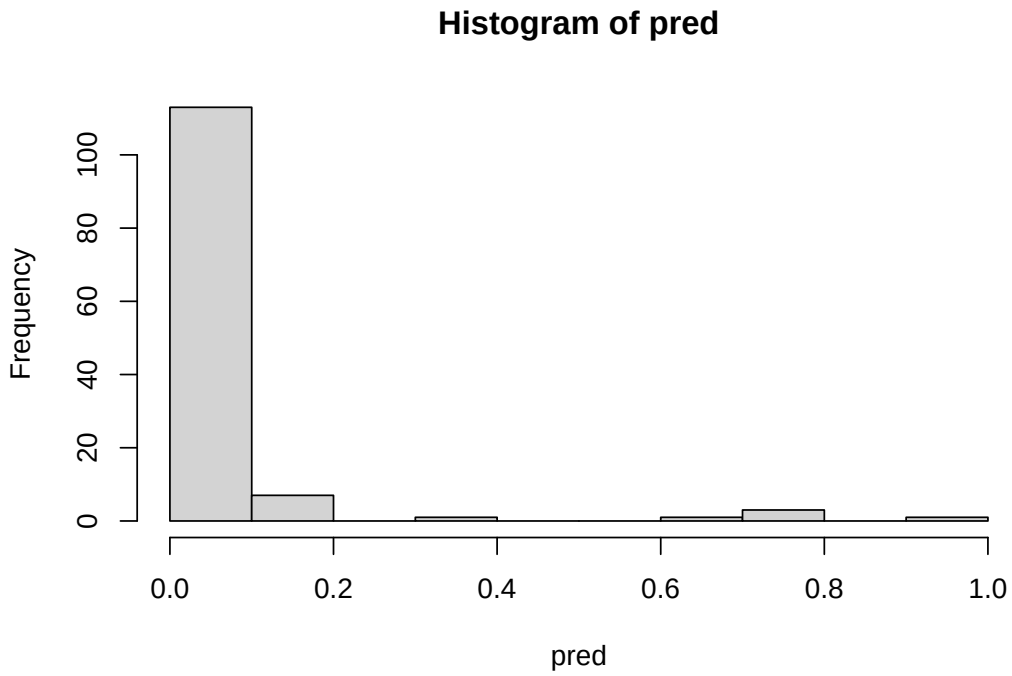
```
## Warning: glm.fit: algorithm did not converge
```

```
## AUC = 1
```

```
##
##           0   1
##   FALSE 457   0
##   TRUE    0  50
```

Logistic Regression with ridge penalty and corss validation.

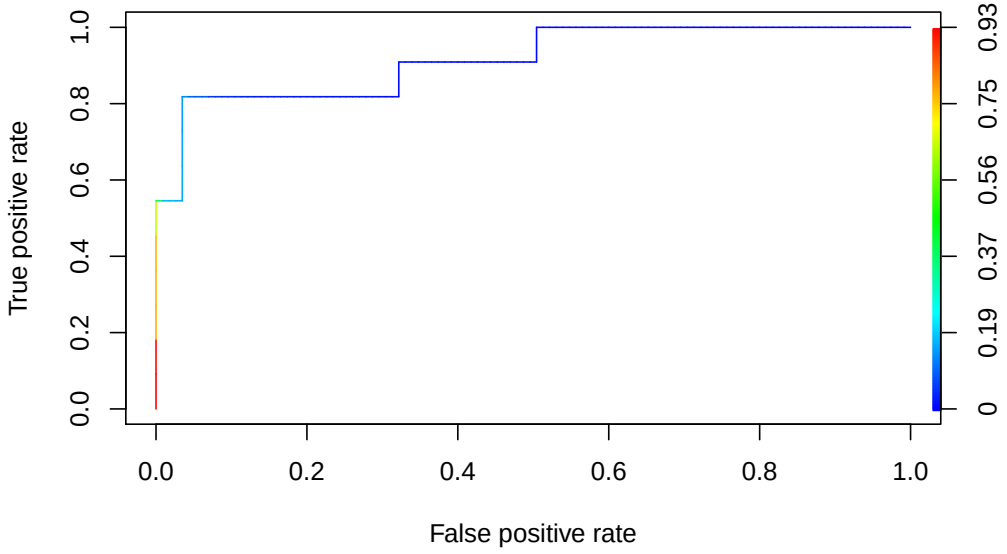
The above model is obviously not correct. Thus, we decided to use penalized regression and cross validation. Thus, we are going to perform a logistic model with ridge penalty and 10-fold cross validation. Since, the Logistic Regresion Model will not return value with 0 and 1 Thus, we decided to choose the cut-off value based on the distribution of the predict result. Thus, we plot the histogram of the prediction result.



Based on the graph shows above, we decided to choose cut-off value as 0.5. The the prediction result will give 121 FALSE and 5 TRUE.

```
##
## FALSE  TRUE
##   121     5
```

Then, we are going to plot the ROC and calculate AUC.



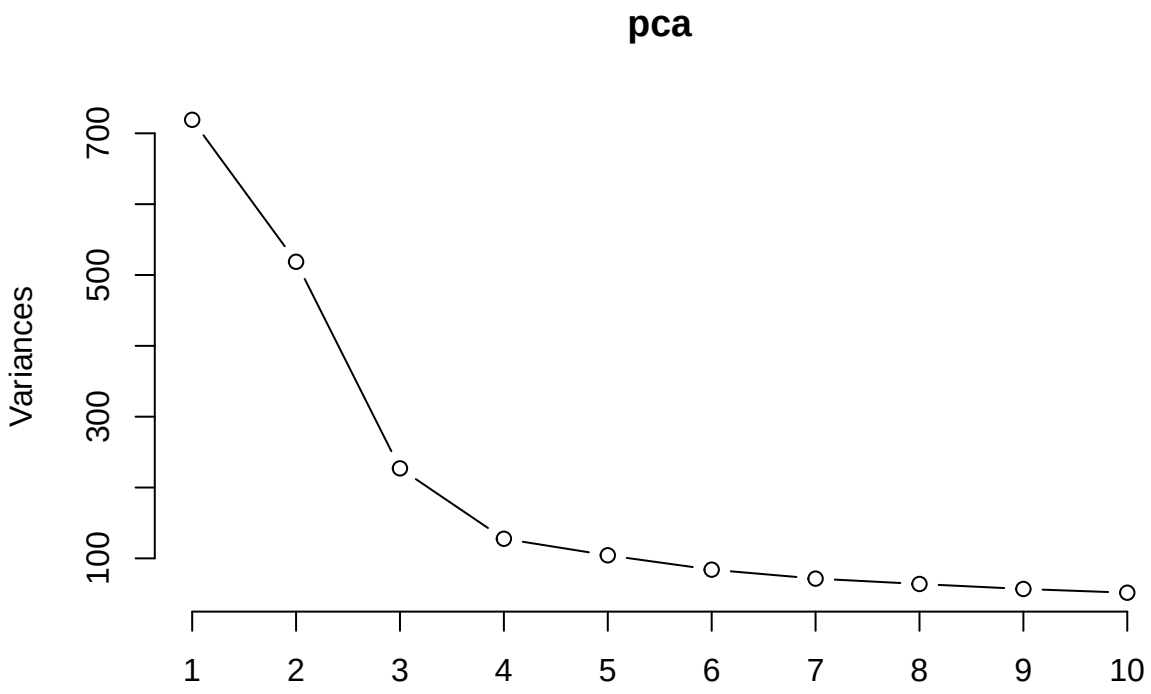

```
## AUC = 0.915415
```

```
##        ytest
##           0   1
##   FALSE 115   6
##   TRUE    0   5
```

Based on the ROC and output given above, AUC is 0.915415, which is a good result

kmean clustering

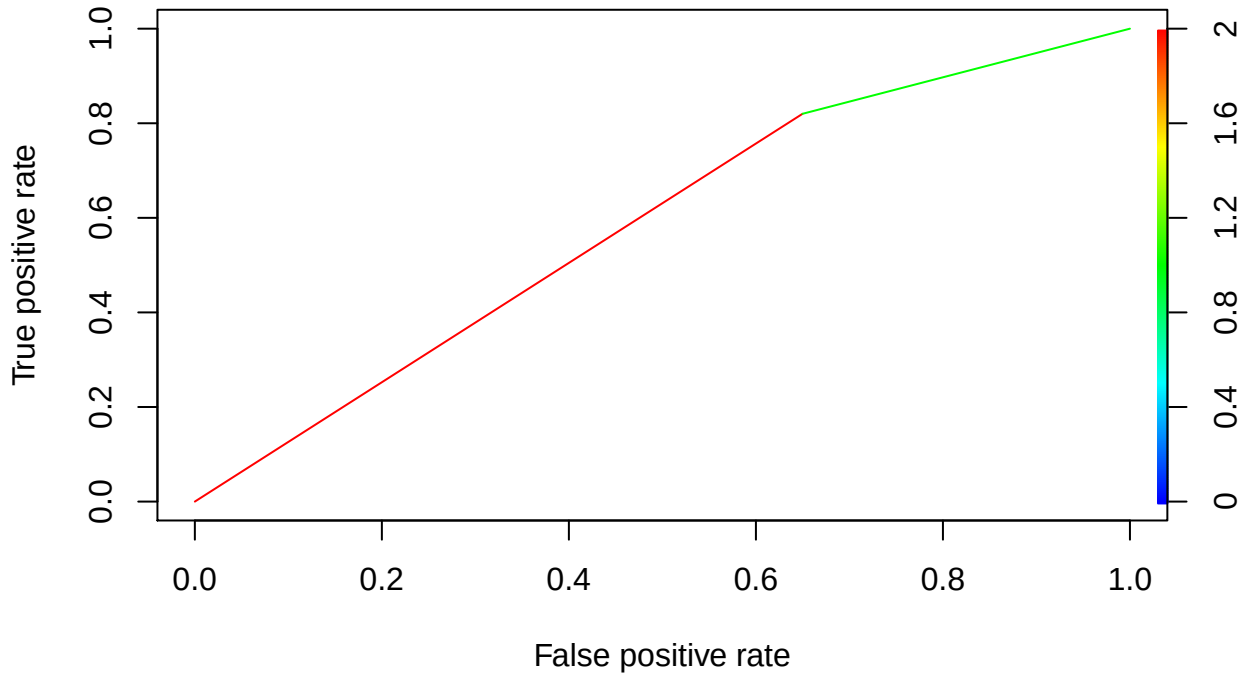
```
# sub4 = sub3[,-1099]
# kmeanfit <- kmeans(sub4, 2)
# table((kmeanfit$cluster - 1),sub3$y)
pca <- prcomp(sub3[, -ncol(sub3)])

plot(pca, type = "l")
```




```
comp = pca$x[,1:4]
kfit = kmeans(comp,2)
clusters = kfit$cluster - 1
table(clusters,sub3$y)
```

```
##
## clusters   0   1
##        0 160   9
##        1 297  41
```

```
roc2 <- prediction(clusters, sub3$y)
# calculates the ROC curve
perf2 <- performance(roc2,"tpr","fpr")
plot(perf2,colorize=TRUE)
```




```
performance(roc2, measure = "auc")@y.values[[1]]
```

```
## [1] 0.5850547
```

tunning tree

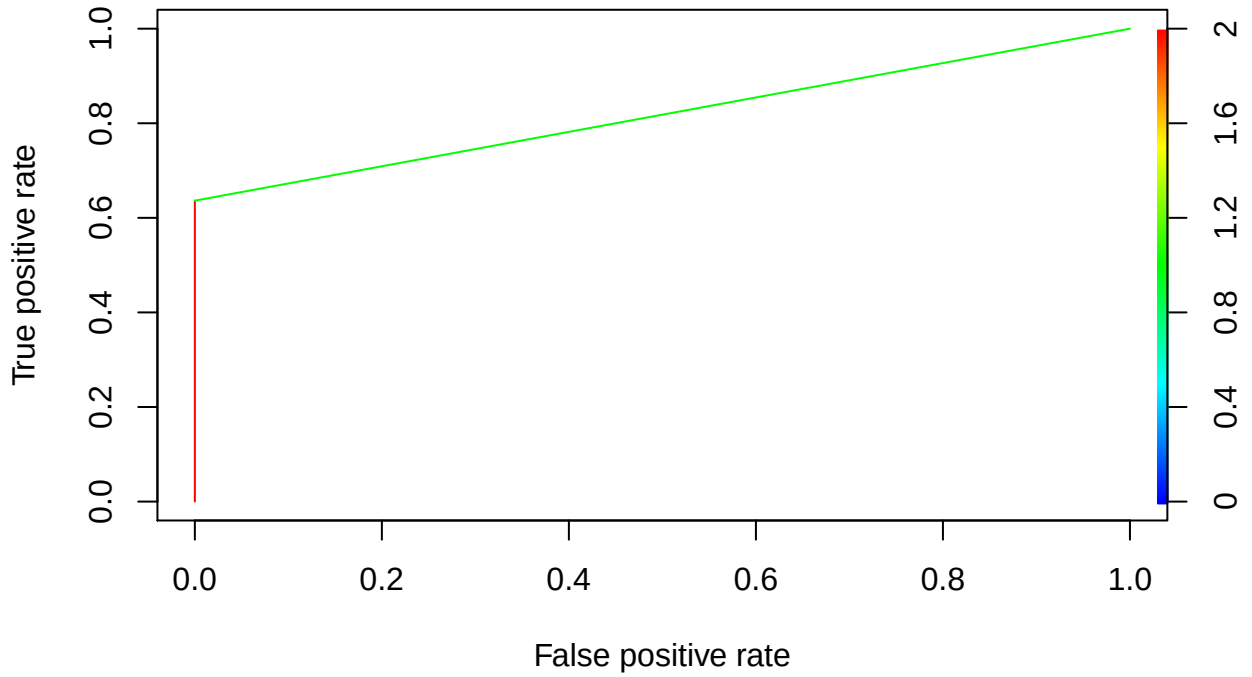
```
library(rpart)
fit = rpart(as.factor(y)~., data= train,control = rpart.control(xval = 10))
fit$cptable
```

```
##      CP nsplit rel error xerror      xstd
## 1 0.44      0      1.00   1.00 0.1342669
## 2 0.08      1      0.56   0.74 0.1171321
## 3 0.06      3      0.40   0.72 0.1156612
## 4 0.01      4      0.34   0.72 0.1156612
```

```
# prunedtree = prune(fit, cp=cptarg)
# rpart.plot(prunedtree)
# table(,ytest)
pred = predict(fit,Xtest)
result = ifelse(pred[,1] > pred[,2], 0, 1)
table(result,ytest)
```

```
##       ytest
## result   0   1
##      0 115   4
##      1   0   7
```

```
roc2 <- prediction(result, ytest)
# calculates the ROC curve
perf2 <- performance(roc2,"tpr","fpr")
plot(perf2,colorize=TRUE)
```




```
performance(roc2, measure = "auc")@y.values[[1]]
```

```
## [1] 0.8181818
```

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.

NOTE: have not cross validated, have not fitted for all outcomes (PR.Status, ER.Status, histological.type, HER2.Final.Status)

```
# fit a random forest with best parameters selected by cross-validation
set.seed(651978735)
rf.fit = randomForest(Xtrain, ytrain,
                      ntree=500,
                      mtry=40,
                      nodesize=15,
                      samplesize=400,
                      importance=TRUE)
```

```
# check the model's test accuracy
pred = predict(rf.fit, Xtest)
confusion_table = table("predicted" = pred, "actual" = ytest)
confusion_table
```

```
##          actual
## predicted   0   1
##         0 115  11
##         1   0   0
```

```
(confusion_table[1, 1] + confusion_table[2, 2]) / test_size
```

```
## [1] 0.9126984
```

```r
# selected the most important 50 variables
impt = rf.fit$importance[order(rf.fit$importance[,3], decreasing=TRUE),][1:50,]
vars = c(rownames(impt), c("pp_E.Cadherin", "pp_PTEN", "mu_TBX3", "cn_FOXA1", "mu_FOXA1", "rs_FOXA1"))
```

```r
# sub4 is the cleaned dataset with all four response variables
# columns: 50 predictors, 4 outcomes
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)
dim(sub4)
```

```
## [1] 507  60
```

```r
# split test and train
Xtest = sub4[test_idx, 1:50]
Xtrain = sub4[-test_idx, 1:50]
```

```r
# fit for ER.Status (SVM)
ytest = sub4$ER.Status[test_idx]
ytrain = sub4$ER.Status[-test_idx]
svm.fit = svm(ytrain ~., data=Xtrain,
              type="C-classification", kernel="linear", scale=F, cost=1)
table("predicted" = svm.fit$fitted, "actual" = ytrain)
```

```
##           actual
## predicted  Negative Positive
##   Negative       73        9
##   Positive       17      282
```

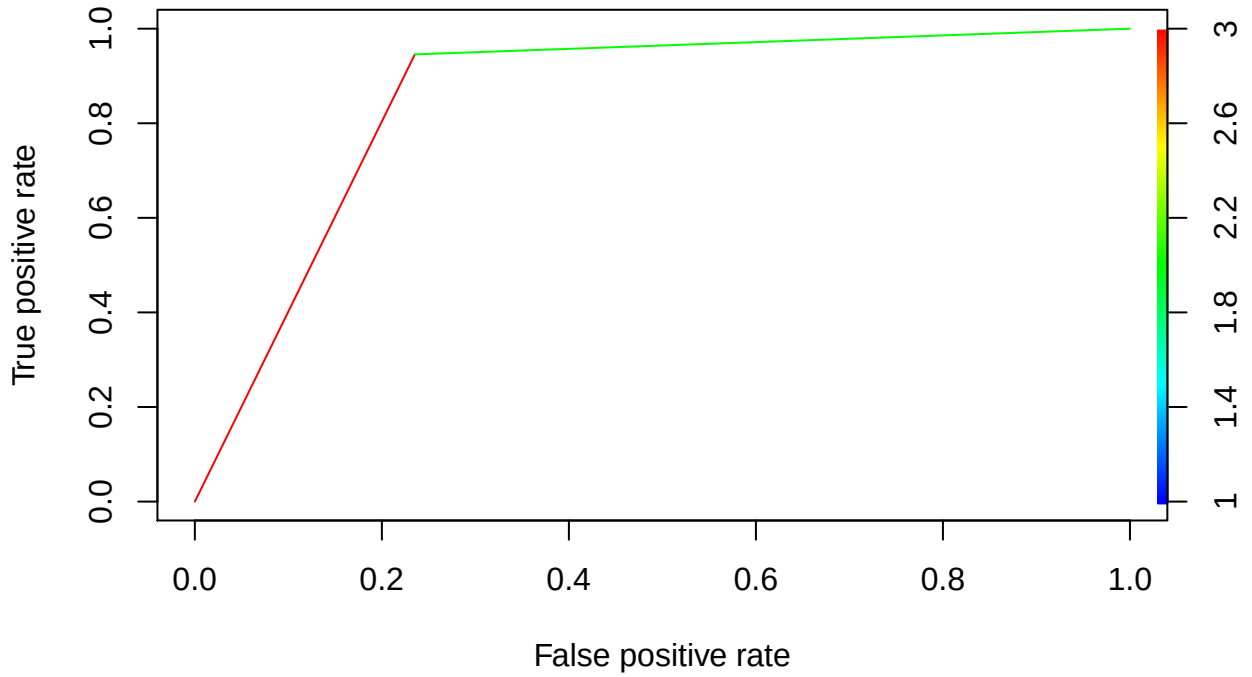
```r
pred = predict(svm.fit, newdata = Xtest)
table("predicted" = pred, "actual" = ytest)
```

```
##           actual
## predicted  Negative Positive
##   Negative       26        5
##   Positive        8       87
```

```r
# the AUC is 0.84
roc = prediction(as.numeric(pred), ytest)
performance(roc, measure = "auc")@y.values[[1]]
```

```
## [1] 0.855179
```

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



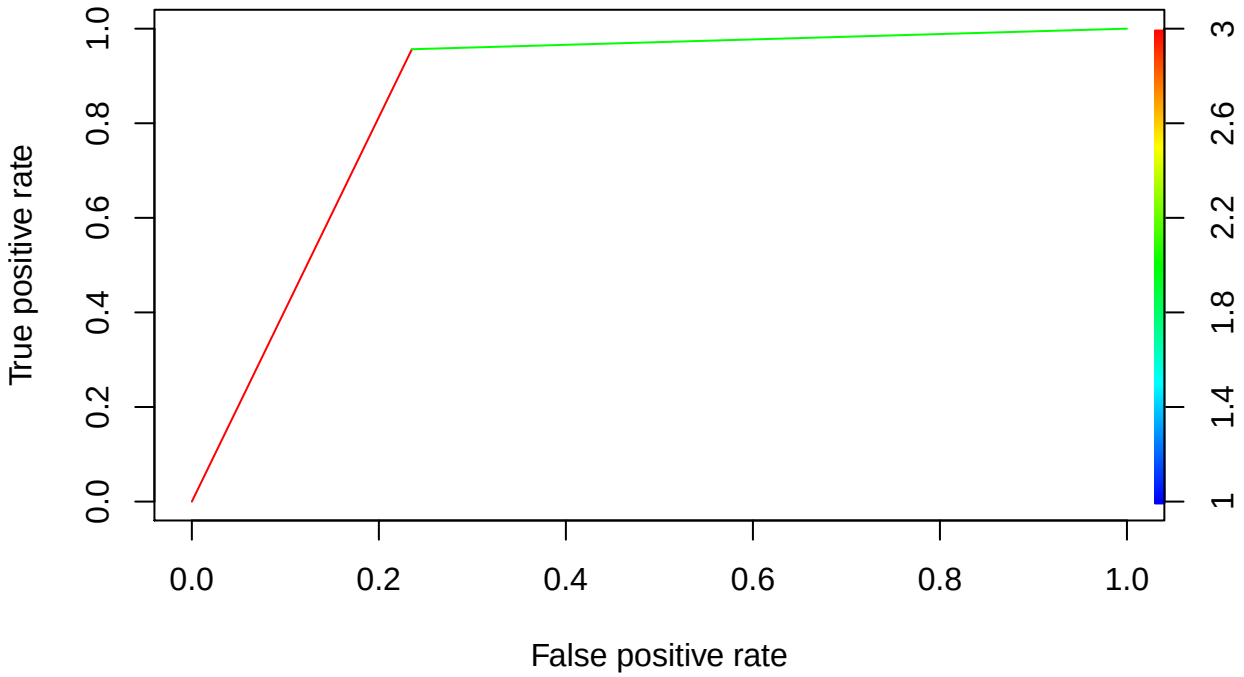

```
# fit for ER.Status, random forest, AUC is 0.90
rf.fit = randomForest(Xtrain, ytrain,
                      ntree=500,
                      mtry=40,
                      nodesize=15,
                      samplesize=400,
                      importance=TRUE)

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

```
##           actual
## predicted  Negative Positive
##   Negative       26        4
##   Positive        8       88
```

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

```
## [1] 0.8606138
```

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




```
set.seed(1)
fold_ID = sample(1:3, 705, replace = TRUE)
fold_id = fold_ID[1:nrow(sub4)]
```