Project

Contents

Note: have not finished data preprocessing

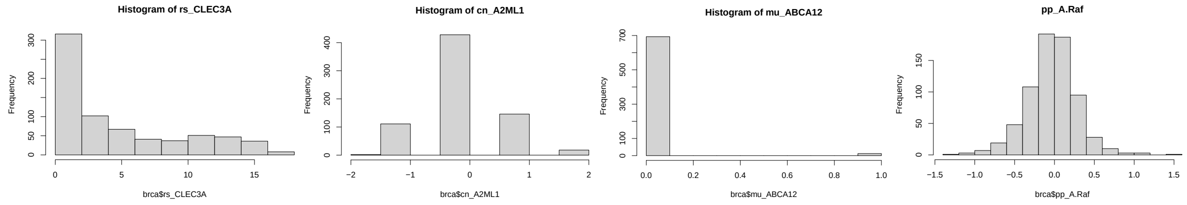
Project Description (est. 1 page, pt. 5)

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

Summary Statistics and Data Preprocessing (est. 1 -2 pages, pt.10)

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.

```r
# sub is the dataset containing no missing values
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
```

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 that belong to 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.

```r
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
```

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

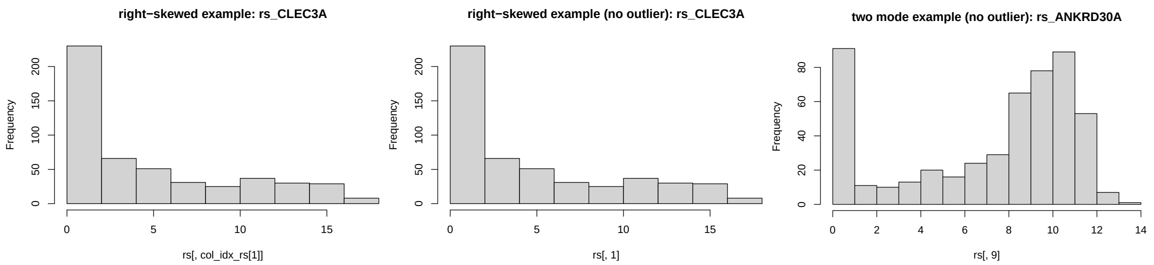
```
## [1] 94
```

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

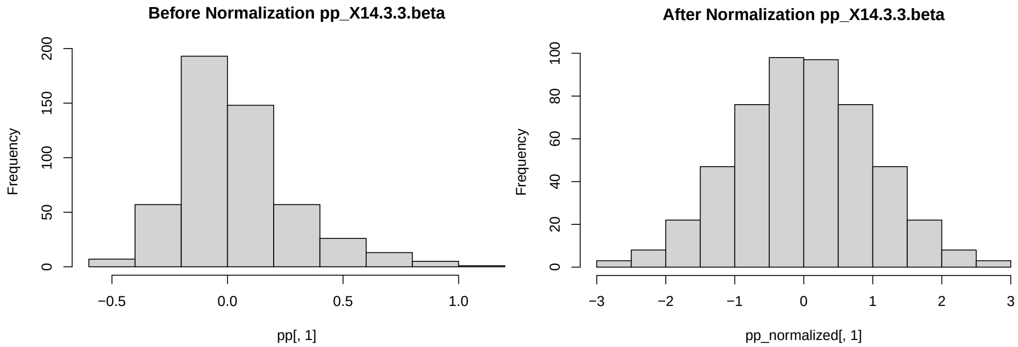
Continuous Predictors

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 variable 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 Predictors

models that perform well on high-dimensional data

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

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

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

```
## [1]  507 1059
```

```
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)
confusion_table = table("fitted" = pred, "actual" = ytest)
confusion_table
```

```
##        actual
## fitted  0  1
##      0 30  8
##      1 13 75
```

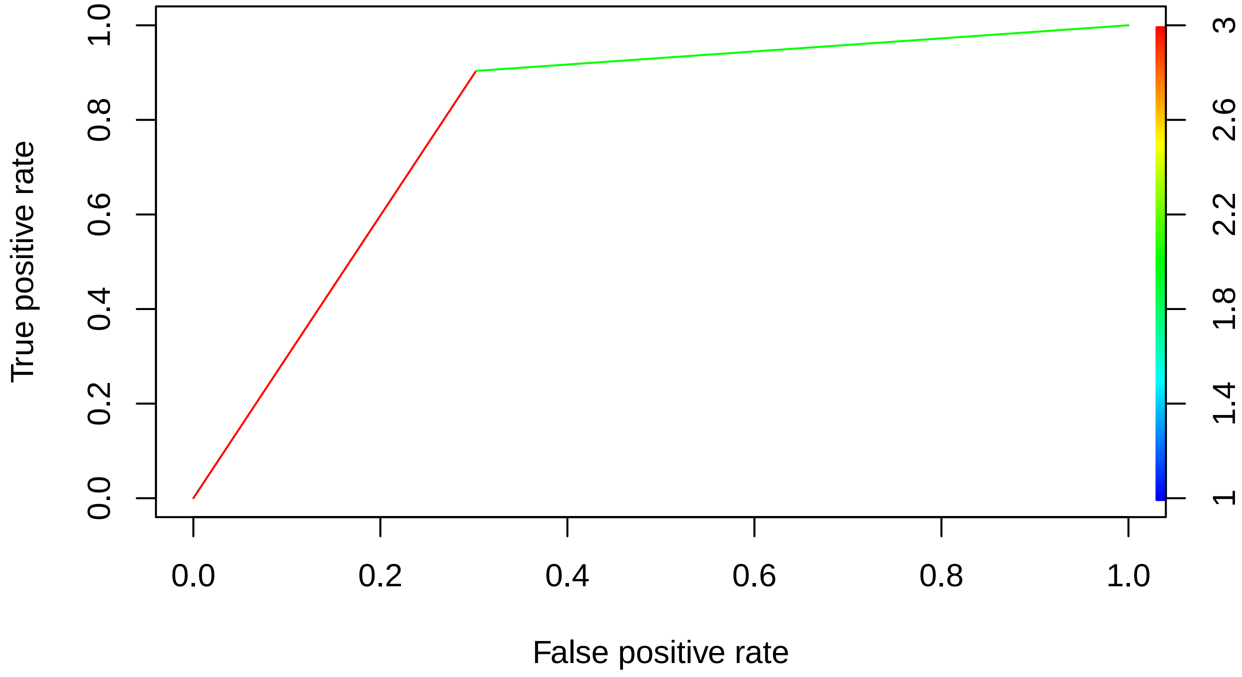
```
# (34 + 69) / (34 + 69 + 14 + 9)
(confusion_table[1, 1] + confusion_table[2, 2]) / test_size
```

```
## [1] 0.8333333
```

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

```
## [1] 0.8006444
```

```
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)
confusion_table = table("fitted" = pred, "actual" = ytest)
confusion_table
```

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

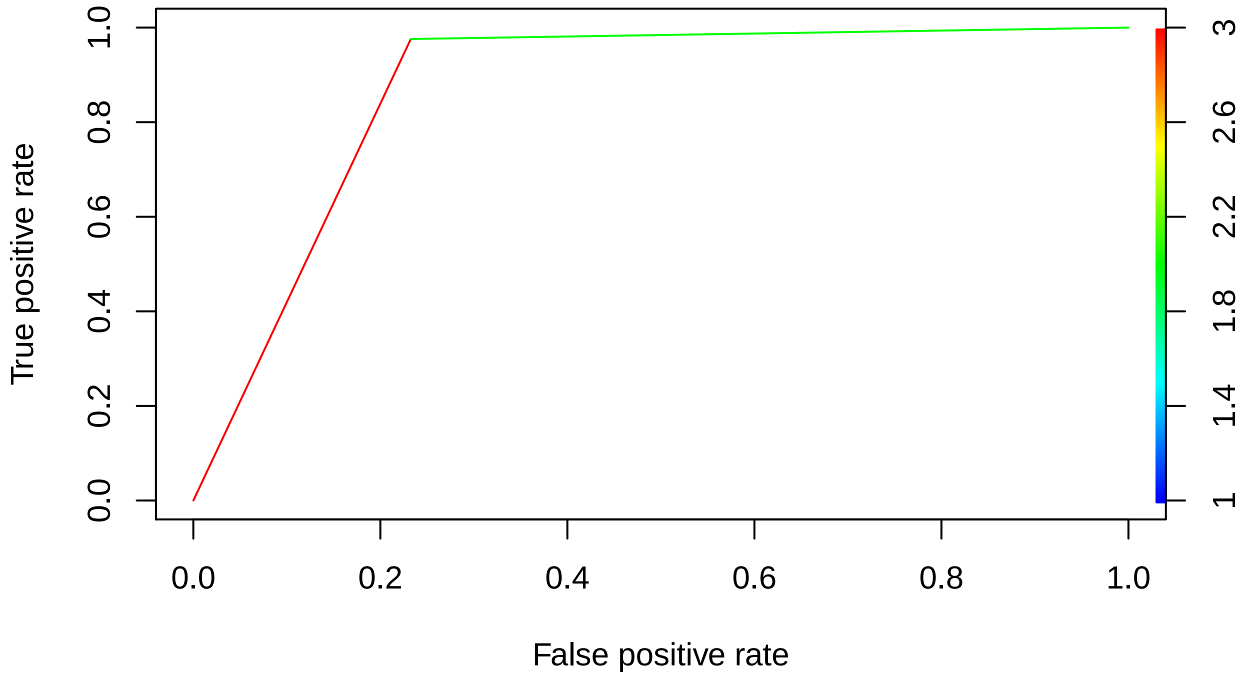
```
(confusion_table[1, 1] + confusion_table[2, 2]) / test_size
```

```
## [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)
```



Histological Type (hcluster and knn regression) (est 2-3 pages, pt.20)

```
y = as.factor(sub$histological.type)
y = ifelse(y == "infiltrating lobular carcinoma", 1, 0)
sub3 = cbind(rs, cn, mu, pp, y) # cleaned dataset with PR.status as response
dim(sub3)
```

```
## [1]  507 1059
```

```
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)]
```

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

```
##
##       0   1
##   0 162   9
##   1 295  41
```

```
acc = (394 + 2)/ nrow(sub4)
acc
```

```
## [1] 0.7810651
```

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)
```

```
Xtest = sub4[test_idx, 1:50]
Xtrain = sub4[-test_idx, 1:50]
```

```
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("fitted" = svm.fit$fitted, "actual" = ytrain)
```

```
##             actual
## fitted      Negative Positive
##   Negative        80        8
##   Positive        10      283
```

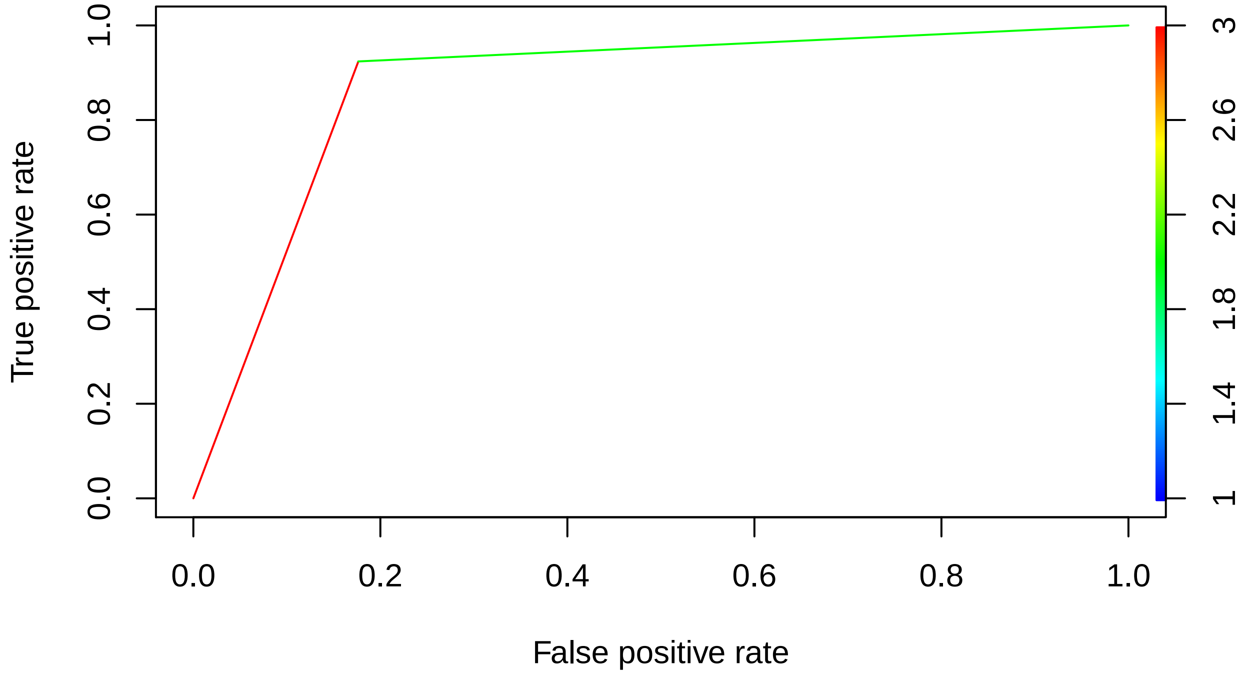
```
pred = predict(svm.fit, newdata = Xtest)
table("fitted" = pred, "actual" = ytest)
```

```
##             actual
## fitted      Negative Positive
##   Negative        28        7
##   Positive         6       85
```

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

```
## [1] 0.8737212
```

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



```
rf.fit = randomForest(Xtrain, ytrain,
                      ntree=500,
                      mtry=7,
                      nodesize=10,
                      samplesize=400,
                      importance=TRUE)

pred = predict(rf.fit, Xtest)
```

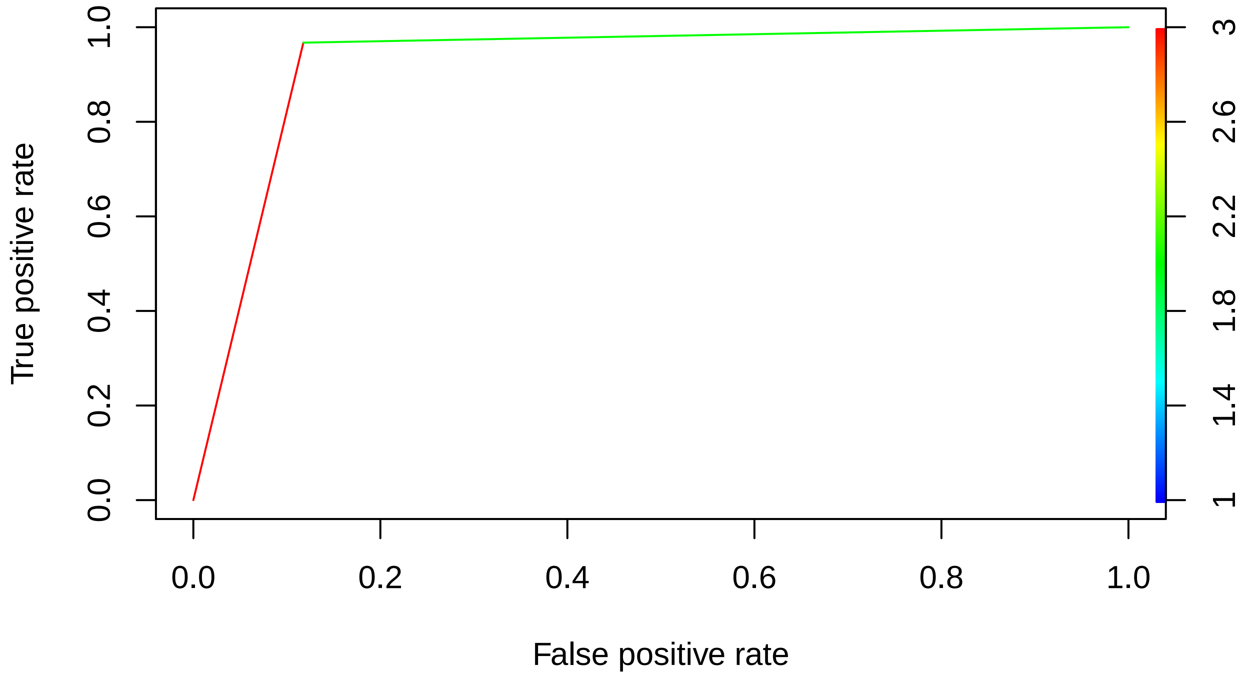
```
table("fitted" = pred, "actual" = ytest)
```

```
##             actual
## fitted      Negative Positive
##   Negative       30        3
##   Positive        4       89
```

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

```
## [1] 0.9248721
```

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


True positive rate
False positive rate