Final Project

Houli Huang

12/3/2021

Contents

1. Project Description and Summary

This project utilizes multiple approaches in modeling Kaggle BRCA Multi-Omics (TCGA) data (loaded as `brca`). The data contains 705 observations, 1936 variables (860 copy number variations, 249 mutations, 604 gene expressions and 223 protein levels), and 5 outcomes (vital.status, PR.Status, ER.Status, HER2.Final.Status, and histological.type). This project focus on modeling 4 of the outcomes: PR.Status, ER.Status, HER2.Final.Status, and histological.type.

Firstly, we applied Univariate Analysis to each variable categories and performed data cleaning and pre-processing based on the result from analysis (Part 3). Some gene expressions variables are selected out, because of the presence of outlier(s).

Secondly, we use 2 different approach on modeling PR.Status (Part 4). The first approach is Penalized Logistic Regression. We use Lasso regression performed by `cv.glmnet`. Two subgroups of variables are selected by this approach. At lambda.min, we have 63 variables. At lambda.1se, we have 4 variables. The second approach is Discriminant Analysis. We tried 4 combinations of settings: both lda and qda with two groups of variables selected by lambda.min and lambda.1se. The best model is Lasso regression with $\lambda =$ `lambda.min`. The 10-fold cross-validated classification error of this model is **0.1391941**.

Thirdly, we use 2 different approach on modeling histological.type (Part 5). The first approach is K-Nearest Neighbors. We also use Lasso regression to reduce input dimension for KNN. The second approach is Support Linear Vector Machine. We use the same group of variables from KNN to train SVM. The best model is linear SVM with `cost` = 0.9526316. And the 10-fold cross validated AUC is **0.94**.

Finally, we need to model 4 outcomes with 50 predictors variables. The first approach we use is Lasso Regression. We fit 4 Lasso Regression models for each outcomes and select desired number of variables based on 3-fold AUC vs number of variables plot. We also tried another approach referenced from "Multiple SVM-RFE for gene selection in cancer classification with expression data.". The mSVM-RFE method can select a small group of important features from very high dimension input (cancer dataset for example). However, when measure by 3-fold AUC, the performance is not as good as Lasso regression. The best model is Lasso regression. And the averaged 3-fold cross validated auc is **0.9257108**

2. Literature Review.

Secondly, we tried servarl models predicting PR.Status and histological.type based on classification measure and auc measure. We used Penalized Logistic Regression and Discriminant Analysis on modeling PR.Status. The Penalized Logistic Regression we use is Lasso regression performed using `cv.glmnet`. We also performed Discrimnant The work from Caihao Cui and Dianhui Wang, suggested using Lasso regression is a good way to reduce input dimension and therefore improve performance of machine learning. Although, we are not going to use machine learning in this project, the idea can be use to improve other models that are sensitive to high dimensions.

3.Summary Statistics and data processing

```
#loading data
brca = read.csv("brca_data_w_subtypes.csv")
```

3.1 data preprocessing

Firstly, we will discard the vital.status variable as required. And transfer all outcomes into binary values. Code omitted.

3.2 Missing Pattern

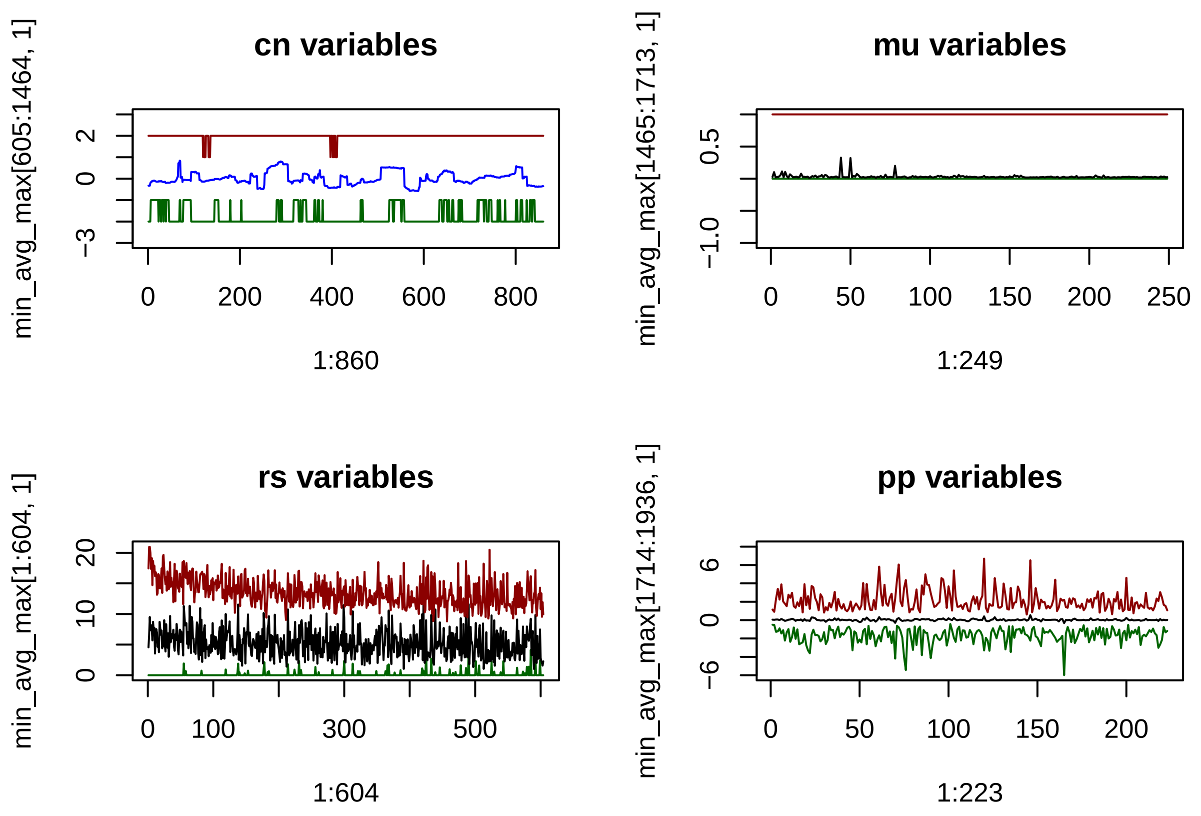
```
library(mice)
miss_pattern = md.pattern(brca[,1:1936], plot = FALSE)
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```

There is no missing values in predictor variables.

3.3 Univariate Analysis

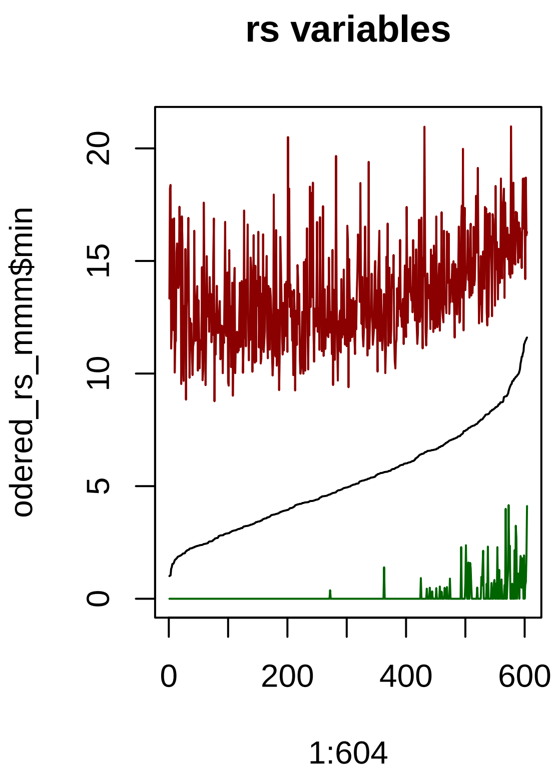
We decide to separate the variables into four groups based on their categories and observe their patterns.

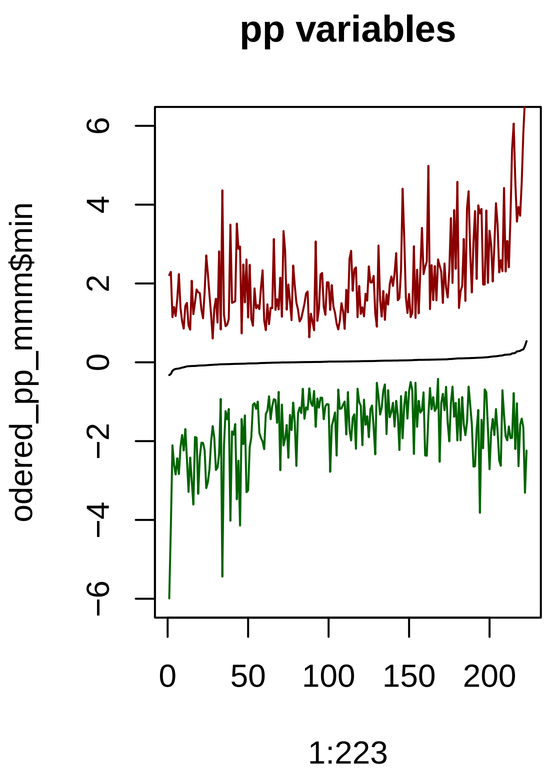


In the above plots, the red curves denote the maximum value of the variable. The green curves denote the minim value of the variable. And the black curves are mean values. We can infer the range of a variable by the space between maximum and minimum.

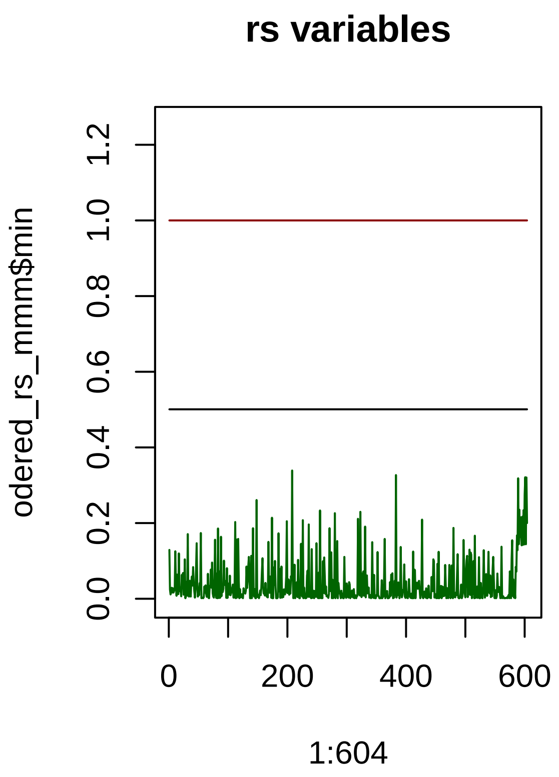
Observations:

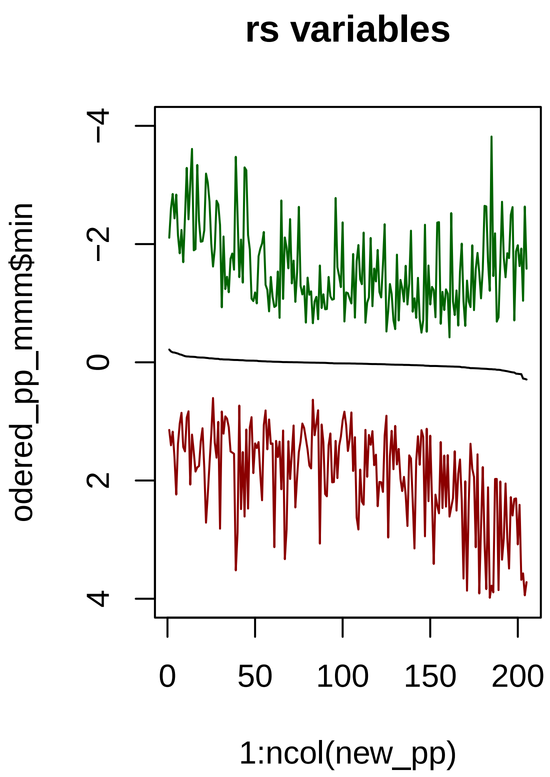
- An interesting finding is that `cn` variables are strictly in the range between -2 and 2. And we find that `cn` variables are discrete. And its values belongs to c(-2,-1,0,1,2). Thus, we can consider transferring `cn` into categorical variables.
- Similarly, `mu` variables have values in c(0,1). However, the data is extremely unbalanced. We need to deal with this issue in the later classification problem.
- Minimum values of rs variables are strictly higher than 0, whereas there is no upper bond for the maximums. This might lead to left-skew distribution.





- After sorting rs and pp by their mean values (see figures above), we find that a rs variable tends to be left-skew if its minimum is 0. For pp variables, we can see the mean values are around 0 for pp variables.A mean value of pp variable below zero is associated with negative outlier(s). And a mean value of pp variable above zero is associated with positive outlier(s).





Solutions:

- categorical transformation on mu variables.
- Apply Quantile transformation on rs variables. (result left figure above).
- Remove pp variables with outlier(s). (result right figure above).

4.Modeling PR.Status

4.1 Preparation before Modeling

```
#discard observations with missing PR.Status value
brca_pr = subset(brca, !is.na(PR.Status))
anyNA(brca_pr$PR.Status)
```

```
## [1] FALSE
```

```
#splitting data into k groups to cross validate
k = 10
group_idx = sample(1:k, nrow(brca_pr), replace = TRUE)
brca_pr$group_idx = group_idx
```

4.2 Lasso Regression Approach

The first approach we decide to use to model `PR.Status` is Lasso Logistic Regression. We notice that there are still many predictor variables left after reprocessing. And penalized logistic regression can select a small set of variables for modeling. This selection will be beneficial to our later study.

We will be using the `cv.glmnet` function in `glmnet` package to perform a Lasso Logistic Regression fit with 10-fold cross validation and using mean classification error as the measure.

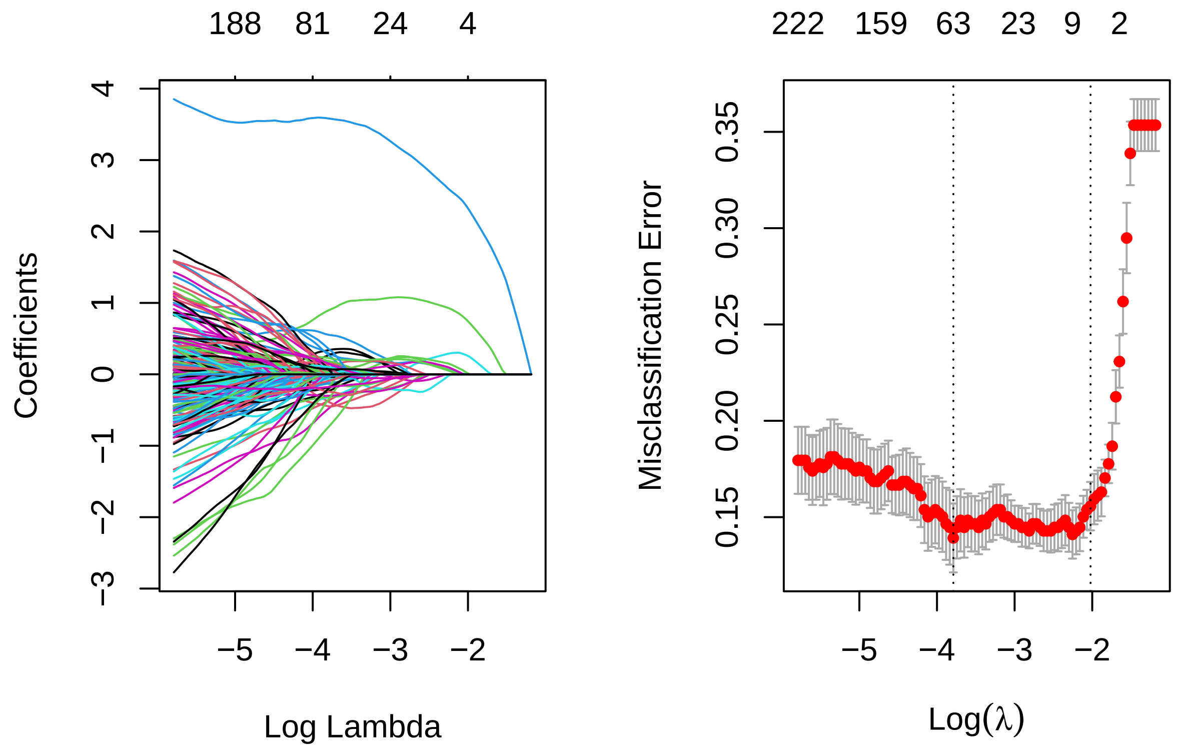
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-2
```

```
## Loading required package: Matrix
## Loaded glmnet 4.1-2
set.seed(3)
lasso.fit = cv.glmnet(x = data.matrix(brca_pr[, 1:1918]), y = brca_pr$PR.Status, nfolds = 10, alpha = 1

par(mfrow=c(1,2))
plot(lasso.fit$glmnet.fit, "lambda")
plot(lasso.fit)
```



The left figure plots the coefficient of each predictor variable in the model against Log(Lambda). We can observe that there are fewer predictor variables selected (with none-zero coefficients) as λ increases. The right figure plots the 10 fold cross-validated classification error against λ. We can see penalized models at `lambda.min` and `lambda.1se` both give good classification errors around 0.15. Interestingly, there are only 4 predictors left when $\lambda =$ `lambda.1se`. This will be very helpful to predict all outcomes with only 50 predictors allowed.

```
lasso.fit$lambda.min
## [1] 0.02261326
lasso.fit$lambda.1se
## [1] 0.1324463
```

Specifically, we have `lambda.min` = 0.02261326 and `lambda.1se` = 0.1324463.

```
lasso.fit$cvm[lasso.fit$lambda == lasso.fit$lambda.min]
## [1] 0.1391941
lasso.fit$cvm[lasso.fit$lambda == lasso.fit$lambda.1se]
## [1] 0.1556777
lasso.fit$nzero[lasso.fit$lambda == lasso.fit$lambda.min]
## s56
##  63
nzero_coef = subset(as.matrix(coef(lasso.fit, s = "lambda.1se")),as.matrix(coef(lasso.fit, s = "lambda.
nzero_coef
##                      s1
## (Intercept) -0.99954587
## rs_CYP2B7P1  0.76668865
## rs_AGR3      0.26859580
## rs_GFRA1     0.02591375
## rs_PGR       2.35892886
```

When $\lambda =$ `lambda.min`, the averaged 10 fold cross-validated classification error is 0.1391941. When $\lambda =$ `lambda.1se`, the averaged 10 fold cross-validated classification error is 0.1556777. When $\lambda =$ `lambda.min`, there are 63 predictors left. When $\lambda =$ `lambda.1se`, the 4 predictors left are rs_CYP2B7P1, rs_AGR3, rs_GFRA1, and rs_PGR.

4.3 Discriminant Analysis

The second approach to modeling PR.Status is Discriminant Analysis. One disadvantage Discriminant Analysis has is it can't take too many input variables. For example, Linear Discriminant Analysis (LDA) requires the number of input variables to be at least lesser than the number of observations. In the case of caner data, we have more than a thousand variables and only 705 observations. To slove this problem, we can use the selected variables from previous part. There are 2 groups of variables, one group selected by `lambda.min` and the other by `lambda.1se`. And we can use both groups on lda and qda, so there will be four combinations.

```
c_err = data.frame(lda.min = NA, lda.1se = NA, qda.min = NA, lda.1se = NA)
library(MASS)
k_err = NULL
selected_var = (as.matrix(coef(lasso.fit, s = "lambda.1se")) != 0)[2:1919]
for (k in 1:10) {
  brca_tra = brca_pr[brca_pr$group_idx != k,]
  brca_tst = brca_pr[brca_pr$group_idx == k,]
  lda.fit = lda(data.matrix(brca_tra[, selected_var]) , brca_tra$PR.Status)
  pred = predict(lda.fit, data.matrix(brca_tst[, selected_var]))
  k_err[k] = mean(pred$class != brca_tst$PR.Status)
}
c_err$lda.1se = mean(k_err)
```

The above procedure is also applied on lda with min predictor group and qda with 1se and min predictor groups. Similar code chunk omitted.

```
c_err
```

```
##   lda.min   lda.1se qda.min lda.1se.1   qda.1se
## 1      NA 0.1412441      NA        NA 0.1626636
```

We find that both lda and qda don't work with variables selected by lambda.min, because there are too many input variables. The best model is lda with formula PR.Status ~ rs_CYP2B7P1 + rs_AGR3 + rs_GFRA1 + rs_PGR. The fact that lda works better than qda suggests our data set is linearly separable. This is a good foundation to apply generalized linear regression.

5. modeling histological.type

5.1 First Approach: KNN

The first approach we utilize is K-Nearest Neighbors. KNN is one of the most common unsupervised statistical training method. And we are starting at full model, i.e. use all predictors to fit the KNN model. the criteria for selecting k is 10-fold AUC.

```
library(caret)
train_control = trainControl(method = "cv", number = 10,
                     classProbs=T,
                     savePredictions = T)
knn.fit = train(histological.type ~. - ER.Status - PR.Status - HER2.Final.Status - group_idx, data = bre
               method = "knn",
               trControl = train_control,
               preProcess = c("center", "scale"),
               tuneLength = 15,
               na.action = na.omit
               )
```

However, we realized that training the full model is over time-consuming and performance is not ideal (with 10-fold AUC around 0.8). We need find a way to reduce dimensionality.

5.2 Reducing Dimensions

Similar to previous parts, we can adapt Lasso Regression to select predictors.

```
library(glmnet)
set.seed(3)
lasso.fit = cv.glmnet(x = data.matrix(brca[, 1:1918]), y = brca$histological.type, nfolds = 10, alpha =
nzero_coef = subset(as.matrix(coef(lasso.fit, s = "lambda.1se")),as.matrix(coef(lasso.fit, s = "lambda.
nzero_coef
```

```
##                          s1
## (Intercept)     -5.07295846
## rs_WNK4          0.05904669
## rs_TMPRSS3       0.09879136
## rs_HPX           0.59226474
## rs_ANKRD43       0.35708115
## rs_LOC389033     0.26556781
## rs_DEGS2         0.07232058
## rs_TNNT3         0.47714041
## mu_CDH1          2.04907668
## pp_beta.Catenin -0.63305764

## Loading required package: ggplot2

## Loading required package: lattice
```

And we will use the selected variables above to fit KNN model.

```
library(MLeval)
knn.fit$bestTune
##    k
## 6 15
x <- evalm(knn.fit, silent = TRUE, showplots = FALSE)
x$stdres$`Group 1`[13,]
##         Score        CI
## AUC-ROC   0.9 0.86-0.94
```

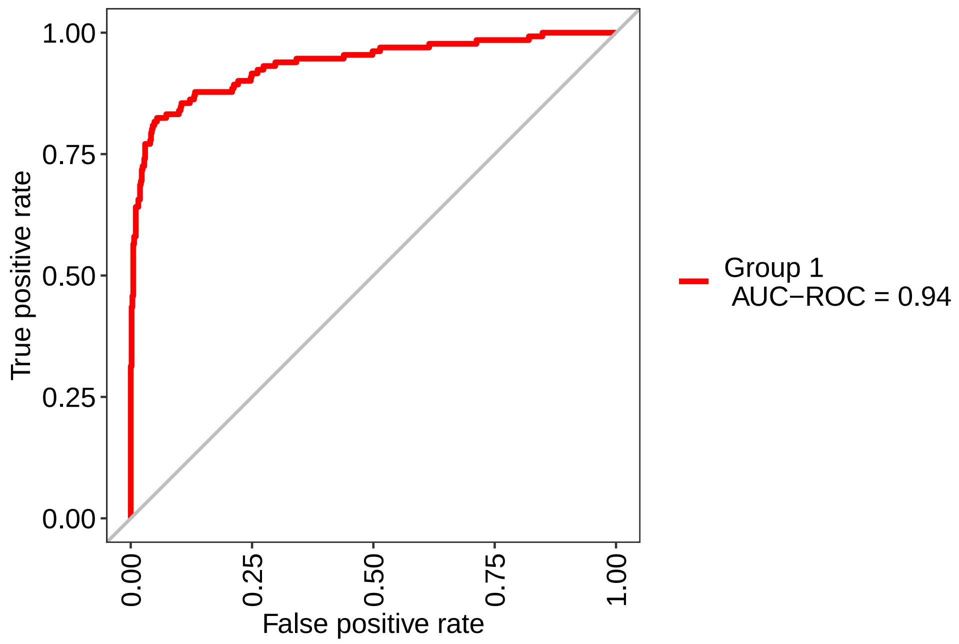
The best tuning is k = 15. And we can find the 10-fold AUC is increased to **0.9**.

5.3 Second Approach: SVM

The second approach we use is linear SVM.

```
cost.grid = expand.grid(cost = seq(0.01, 2, length = 20))
train_control = trainControl(method = "cv", number = 10,
                     classProbs=T,
                     savePredictions = T)
svm.fit = train(histological.type ~rs_WNK4 + rs_TMPRSS3 + rs_HPX+rs_ANKRD43+rs_LOC389033 + rs_DEGS2+rs_
                trControl = train_control,
                tuneGrid = cost.grid,
                na.action = na.omit)
```

```
svm.fit$bestTune
##        cost
## 3 0.2194737
x <- evalm(svm.fit, silent = TRUE, showplots = FALSE)
x$stdres$`Group 1`[13,]
##         Score        CI
## AUC-ROC  0.94 0.91-0.97
x$roc
```



The best tuning is cost = 0.2194737. The 10-fold cross validated AUC is **0.94**, higher than KNN model. **The final model we use to model histological.type is linear SVM with formula (histological.type ~rs_WNK4 + rs_TMPRSS3 + rs_HPX+rs_ANKRD43+rs_LOC389033 + rs_DEGS2+rs_TNNT3+mu_CDH1+pp_beta.Catenin) and cost = 0.2194737.**

6. Predict all outcomes

```
set.seed(1)
foldID = sample(1:3, 705, replace = TRUE)
auc_score = data.frame(matrix(0,ncol = 5, nrow = 2))
colnames(auc_score) = c("PR.Status","ER.Status","HER2.Final.Status","histological.type", "ALL")
row.names(auc_score) = c("auc", "nzero")
```

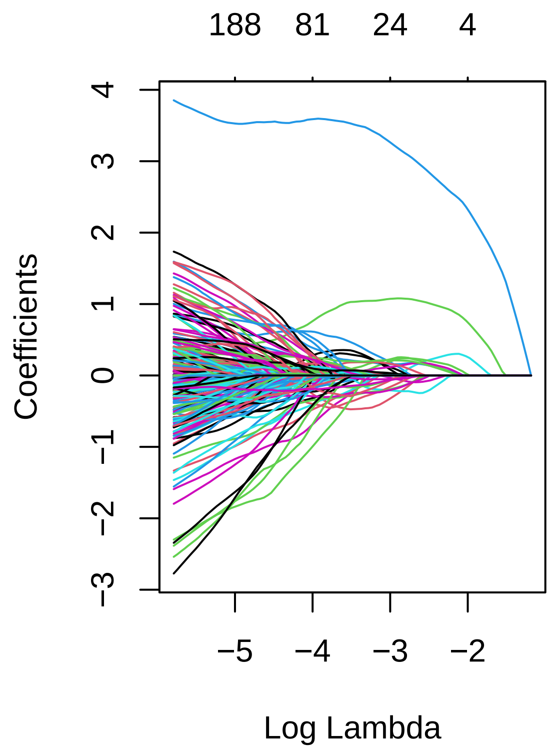
Firstly, we need to assign foldid to each observations. And we also set up a data.frame `auc_score` to record the 3-fold cross-validated AUC of each outcome variable.

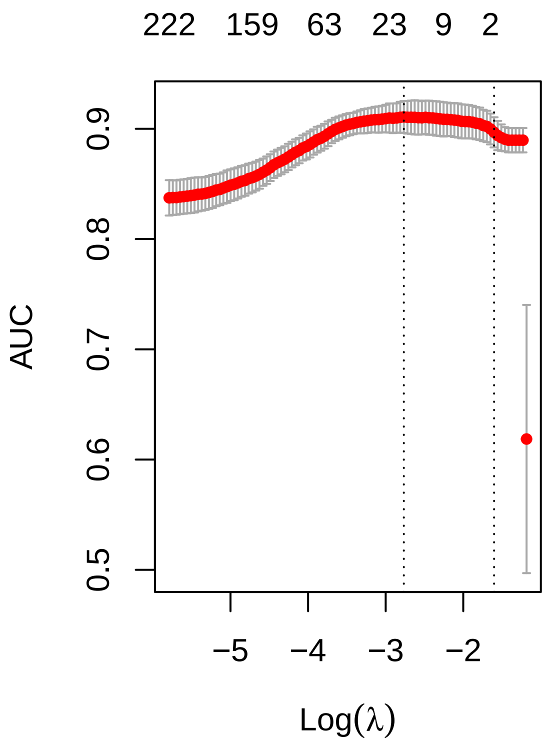
6.1 Use Lasso Regression to Select Predictors for Each Outcome

We can use Logistic Lasso Regression to select a small group of variables (same as we do previously). Conveniently, we can directly supply the foldid (set up above) to the `cv.glmnet` function. Like we did before, use AUC measure for model selection.

```
lasso.fit = cv.glmnet(x = data.matrix(brca[!is.na(brca$PR.Status), 1:1918]), y = brca[!is.na(brca$PR.Sta
nzero_coef = subset(as.matrix(coef(lasso.fit, s = "lambda.min")),as.matrix(coef(lasso.fit, s = "lambda.
coef_pr = nzero_coef
auc_score[1, 1] = lasso.fit$cvm[lasso.fit$lambda == lasso.fit$lambda.min]
auc_score$PR.Status #print AUC score
## [1] 0.9107341 0.0000000
auc_score[2, 1] = lasso.fit$nzero[lasso.fit$lambda == lasso.fit$lambda.min] #number of none zero coeffi
```

```r
par(mfrow=c(1,2))
plot(lasso.fit$glmnet.fit, "lambda")
plot(lasso.fit)
```





When $\lambda =$ `lambda.min`, there are 18 predictor variables selected for modeling PR.Status. And the AUC is 0.9107341. The overall performance is good. And we are going to use this same approach on predicting the other 3 outcomes.

```
##      PR.Status   ER.Status HER2.Final.Status histological.type
## 1  (Intercept) (Intercept)       (Intercept)       (Intercept)
## 2  rs_CYP2B7P1 rs_CYP2B7P1         rs_PPP4R4             rs_C7
## 3  rs_SERPINA6     rs_AGR3          rs_NPY5R           rs_WNK4
## 4      rs_AGR3     rs_ESR1        cn_PPP1R1B       rs_TMEM132C
## 5     rs_FABP7        <NA>          cn_IKZF3    rs_LOC100271831
## 6     rs_GFRA1        <NA>           cn_HAP1        rs_TMPRSS3
## 7       rs_PGR        <NA>           cn_RND2            rs_HPX
## 8     rs_A2ML1        <NA>            cn_PYY           rs_DLX2
## 9      rs_GRPR        <NA>           cn_GFAP          rs_RERGL
## 10     rs_NAT1        <NA>         cn_KIF18B        rs_ANKRD43
## 11 rs_PPP1R14C        <NA>          cn_CRHR1          rs_FXYD1
## 12    rs_RGS22        <NA>           cn_MAPT      rs_LOC389033
## 13   rs_TUBA3E        <NA>              <NA>          rs_DEGS2
## 14     rs_SBSN        <NA>              <NA>          rs_TNNT3
## 15    rs_SOX11        <NA>              <NA>          cn_RIMS2
## 16   rs_RASAL1        <NA>              <NA>          cn_BCAS1
## 17    cn_F2RL2        <NA>              <NA>           mu_CDH1
## 18    cn_EDIL3        <NA>              <NA>         pp_SLC1A5
## 19     pp_ASNS        <NA>              <NA>   pp_beta.Catenin
```

After modeling the other 3 outcomes, we will use the AUC vs. log(lambda) plots to pick λ value to select a small group of predictors for each outcome. The table above lists the predictor variables selected for each outcome. We use $\lambda =$ `lambda.min` for PR.Status and HER2.Final.Status, `lambda.1se` for histological.type, and a λ between `lambda.min` and `lambda.1se` for ER.Status. As result, 18 variables are selected for PR.Status, 3 for ER.Status, 11 for HER2.Final.Status, and 18 for histological.type. The total number of variables we use is **50**.

```
##        PR.Status ER.Status HER2.Final.Status histological.type        ALL
## auc    0.9107341 0.9491024         0.9106935         0.9323132  0.9257108
## nzero 18.0000000 3.0000000        11.0000000        18.0000000 50.0000000
```

The table above lists 3-fold AUC for each outcome and number of predictors selected. **The averaged 3-fold AUC across all outcomes is 0.9257108**

6.2 mSVM-RFE

Another model selection method is reference from ***Multiple SVM-RFE for gene selection in cancer classification with expression data*** (Duan, Kai-Bo et al). mSVM-RFE is a feature selection method that can cut variables by half each iteration (instead of one by one). This function is very convenient in our case with 1918 variables. The usage of svmRFE function is in reference to demo.R [https://github.com/johncolby/SVM-RFE].

```
set.seed(12345)
library(e1071)
source('msvmRFE.R')
input = cbind.data.frame(brca$histological.type, brca[,1:1918])
input[,1466:1714] = as.numeric(input[,1466:1714] == 1)
x = svmRFE(input, k=10, halve.above=100)
```

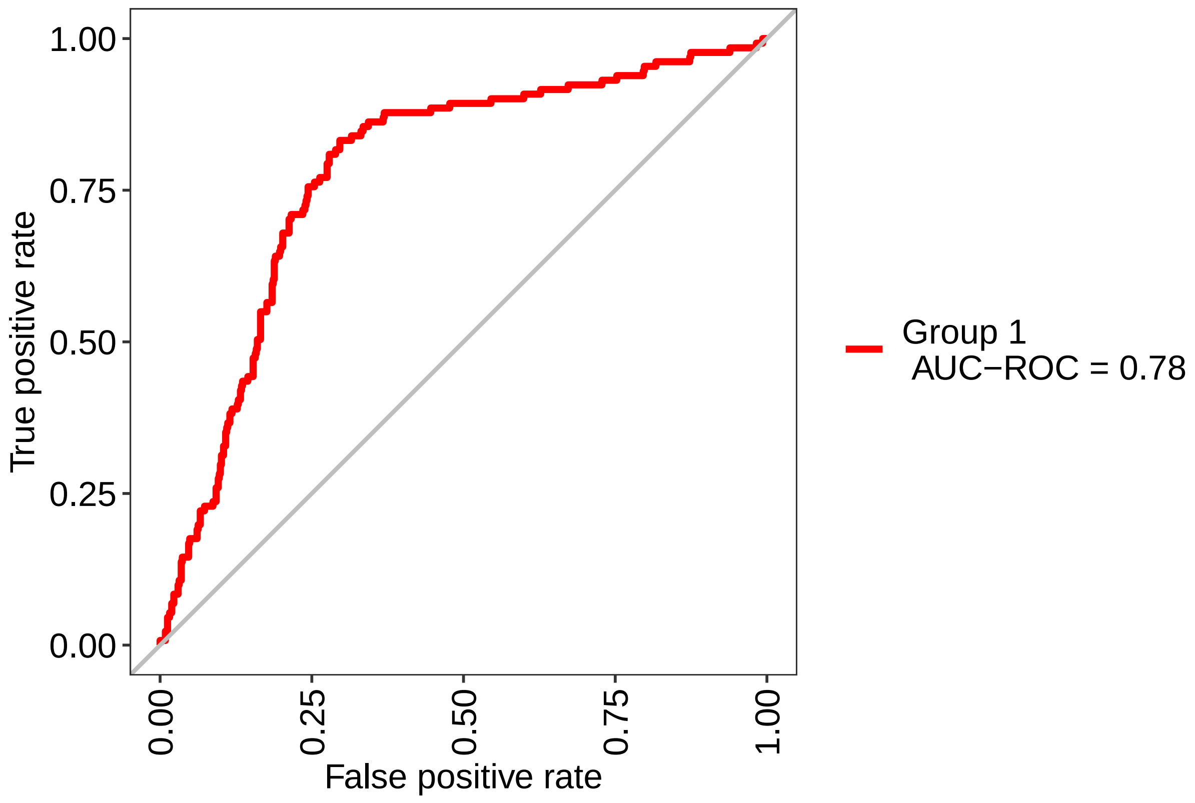
x is a vector where each element is the rank of importance of the corresponding feature.

```
var_name = colnames(brca)[2:1918]
var_name = var_name[x < 20]
var_name
```

```
##  [1] "rs_AGR3"             "rs_FABP4"            "rs_PCK1"
##  [4] "rs_F7"               "cn_KANK4"            "cn_EDIL3"
##  [7] "cn_UNC5A"            "cn_GATA4"            "cn_PI15"
## [10] "cn_ITIH5"            "cn_CXCR5"            "cn_FUT6"
## [13] "cn_CAPS"             "cn_KLK11"            "cn_BMP7"
## [16] "mu_HRNR"             "mu_PCNT"             "mu_SRCAP"
## [19] "pp_X14.3.3.epsilon"
```

Above is the top 20 most important features selected.

```
x <- evalm(svm.fit, silent = TRUE, showplots = FALSE)
x$roc
```



After fitting a linear svm model with the above predictors, we have auc = 0.79. The AUC score is no as good as previous approach. We decided to keep the Lasso model in part 6.2.

7. Reference

1. Duan, Kai-Bo et al. "Multiple SVM-RFE for gene selection in cancer classification with expression data." IEEE transactions on nanobioscience vol. 4,3 (2005): 228-34. doi:10.1109/tnb.2005.853657

2. Caihao Cui, Dianhui Wang, "High dimensional data regression using Lasso model and neural networks with random weights." Information Sciences, Volume 372, 2016, Pages 505-517, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2016.08.060. (https://www.sciencedirect.com/science/article/pii/S0020025516306314)

3.