CS6140 Assignment 3

Chengbo Gu

1. Data preparation and exploration

(a) Select the training set

I downloaded the data from the website and read it using read.table(). Then I used sample function to partition the dataset into a training set and a validatoin set of equal size. Notice that there is a categorical feature famhist, I set "Present" to be 1 and "Absent" to be 0 accordingly.

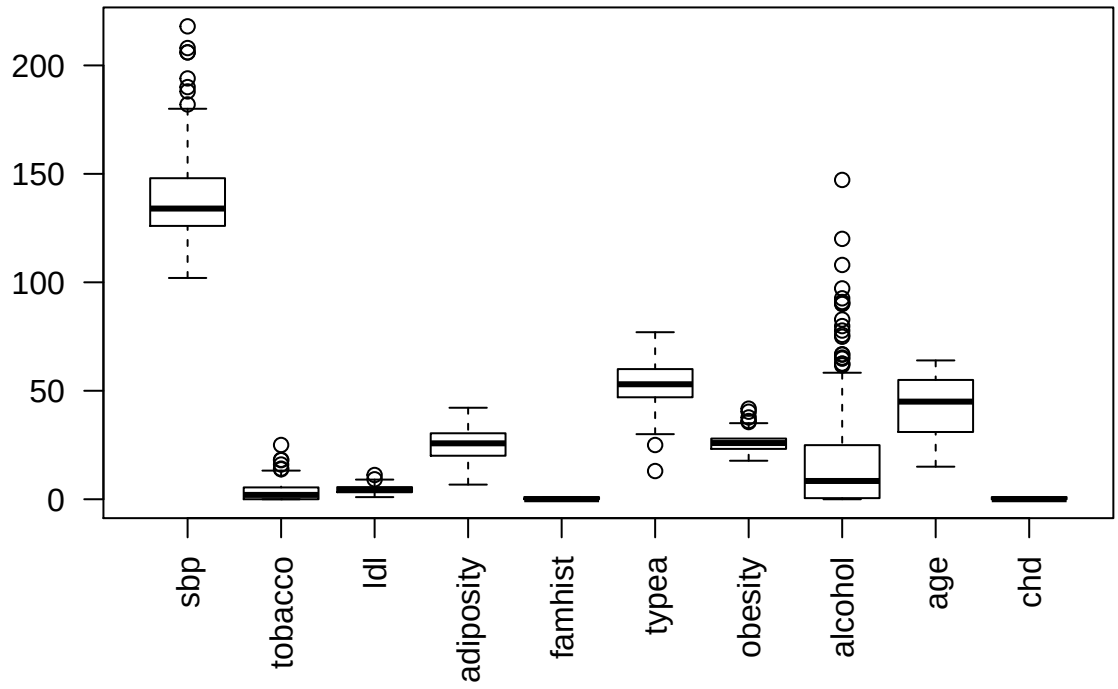
```
set.seed(123)
Mydata <- read.table("SouthAfricanHeartDisease.txt", sep=",",
                     stringsAsFactors = FALSE, header = TRUE)
Mydata <- Mydata[,-1]
Mydata[,5][Mydata[,5]=="Present"] <- 1
Mydata[,5][Mydata[,5]=="Absent"] <- 0
Mydata[,5] <- as.integer(Mydata[,5])
predictors <- Mydata[,1:9]
response <- Mydata[,10]
train <- sample(x=1:nrow(Mydata), size=nrow(Mydata)/2)
trainSet <- Mydata[train,]
testSet <- Mydata[-train,]
```

(b) Data exploration

```
options(width=100)
# missing values
sum(is.na(Mydata))
```

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

```
# outliers
boxplot(trainSet, las = 2)
```



As we can see above, there is no missing value in our dataset. Besides, the boxplot indicates that there seems to be some outliers according to sbp and alchohol. However, different from the rest doesn't necessarily mean wrong as we discussed in class. Since data is scarce, we decided not to remove these "outliers". After this, we can apply one-variable and two-variable summary.

one variable summary

simple one variable summary

```
# One variable summary
one.variable.summary <- summary(trainSet, digits=2)
one.variable.summary
```

```
##       sbp         tobacco          ldl          adiposity      famhist        typea
##  Min.   :102   Min.   : 0.0   Min.   : 0.98   Min.   : 6.7   Min.   :0.00   Min.   :13
##  1st Qu.:126   1st Qu.: 0.0   1st Qu.: 3.23   1st Qu.:20.1   1st Qu.:0.00   1st Qu.:47
##  Median :134   Median : 2.0   Median : 4.33   Median :25.8   Median :0.00   Median :53
##  Mean   :139   Mean   : 3.5   Mean   : 4.57   Mean   :25.2   Mean   :0.43   Mean   :53
##  3rd Qu.:148   3rd Qu.: 5.4   3rd Qu.: 5.59   3rd Qu.:30.4   3rd Qu.:1.00   3rd Qu.:60
##  Max.   :218   Max.   :25.0   Max.   :11.17   Max.   :42.2   Max.   :1.00   Max.   :77
##     obesity      alcohol           age          chd
##  Min.   :18   Min.   :  0.00   Min.   :15   Min.   :0.00
##  1st Qu.:23   1st Qu.:  0.51   1st Qu.:31   1st Qu.:0.00
##  Median :26   Median :  8.42   Median :45   Median :0.00
##  Mean   :26   Mean   : 18.52   Mean   :43   Mean   :0.35
##  3rd Qu.:28   3rd Qu.: 24.95   3rd Qu.:55   3rd Qu.:1.00
##  Max.   :42   Max.   :147.19   Max.   :64   Max.   :1.00
```

We could see more details using the following commands.

```
format(round(stat.desc(trainSet, basic=F), 2), nsmall = 2)
```

```
##                 sbp tobacco  ldl adiposity famhist typea obesity alcohol    age  chd
## median       134.00    2.00 4.33     25.78    0.00 53.00   25.91    8.42  45.00 0.00
## mean         138.62    3.46 4.57     25.21    0.43 53.47   25.92   18.52  42.51 0.35
## SE.mean        1.33    0.27 0.12      0.50    0.03  0.64    0.25    1.67   0.95 0.03
## CI.mean.0.95   2.62    0.54 0.23      0.98    0.06  1.26    0.50    3.28   1.87 0.06
## var          407.58   17.21 3.16     56.69    0.25 94.98   14.79  642.03 208.50 0.23
## std.dev       20.19    4.15 1.78      7.53    0.50  9.75    3.85   25.34  14.44 0.48
## coef.var       0.15    1.20 0.39      0.30    1.16  0.18    0.15    1.37   0.34 1.35
```

```
format(round(stat.desc(trainSet, desc=F), 2), nsmall=2)
```

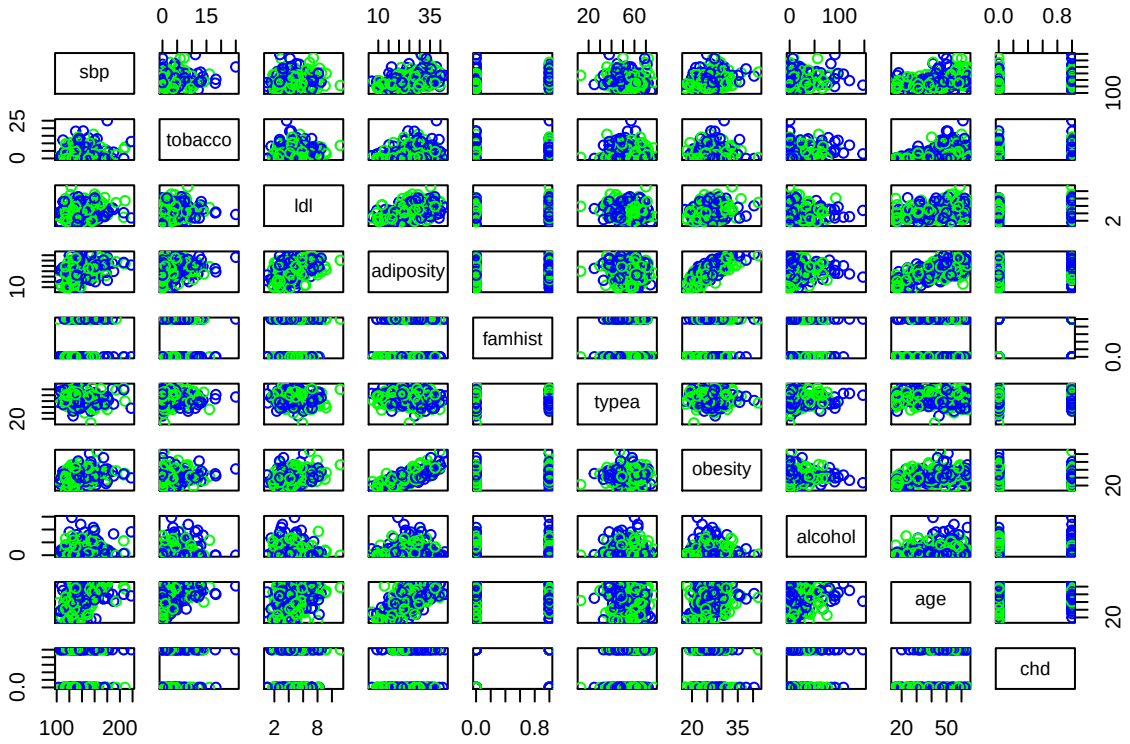
```
##               sbp tobacco     ldl adiposity famhist    typea obesity alcohol     age    chd
## nbr.val    231.00  231.00  231.00    231.00  231.00   231.00  231.00  231.00  231.00 231.00
## nbr.null     0.00   59.00    0.00      0.00  132.00     0.00    0.00   56.00    0.00 149.00
## nbr.na       0.00    0.00    0.00      0.00    0.00     0.00    0.00    0.00    0.00   0.00
## min        102.00    0.00    0.98      6.74    0.00    13.00   17.75    0.00   15.00   0.00
## max        218.00   25.01   11.17     42.17    1.00    77.00   41.76  147.19   64.00   1.00
## range      116.00   25.01   10.19     35.43    1.00    64.00   24.01  147.19   49.00   1.00
## sum      32021.00  799.44 1056.75   5824.01   99.00 12352.00 5988.62 4278.03 9819.00  82.00
```

two variable summary

```
# Two variable summary
round(cor(trainSet), digits=2)
```

```
##              sbp tobacco   ldl adiposity famhist typea obesity alcohol   age  chd
## sbp         1.00    0.17  0.14      0.34    0.11 -0.07    0.25    0.18  0.38 0.17
## tobacco     0.17    1.00  0.17      0.29    0.17 -0.01    0.10    0.23  0.46 0.29
## ldl         0.14    0.17  1.00      0.43    0.21 -0.05    0.34   -0.09  0.33 0.24
## adiposity   0.34    0.29  0.43      1.00    0.17 -0.12    0.77    0.08  0.67 0.23
## famhist     0.11    0.17  0.21      0.17    1.00 -0.01    0.11    0.08  0.28 0.33
## typea      -0.07   -0.01 -0.05     -0.12   -0.01  1.00    0.00    0.02 -0.12 0.12
## obesity     0.25    0.10  0.34      0.77    0.11  0.00    1.00    0.01  0.32 0.10
## alcohol     0.18    0.23 -0.09      0.08    0.08  0.02    0.01    1.00  0.17 0.12
## age         0.38    0.46  0.33      0.67    0.28 -0.12    0.32    0.17  1.00 0.35
## chd         0.17    0.29  0.24      0.23    0.33  0.12    0.10    0.12  0.35 1.00
```

```
pairs(trainSet, col=c("green", "blue"))
```



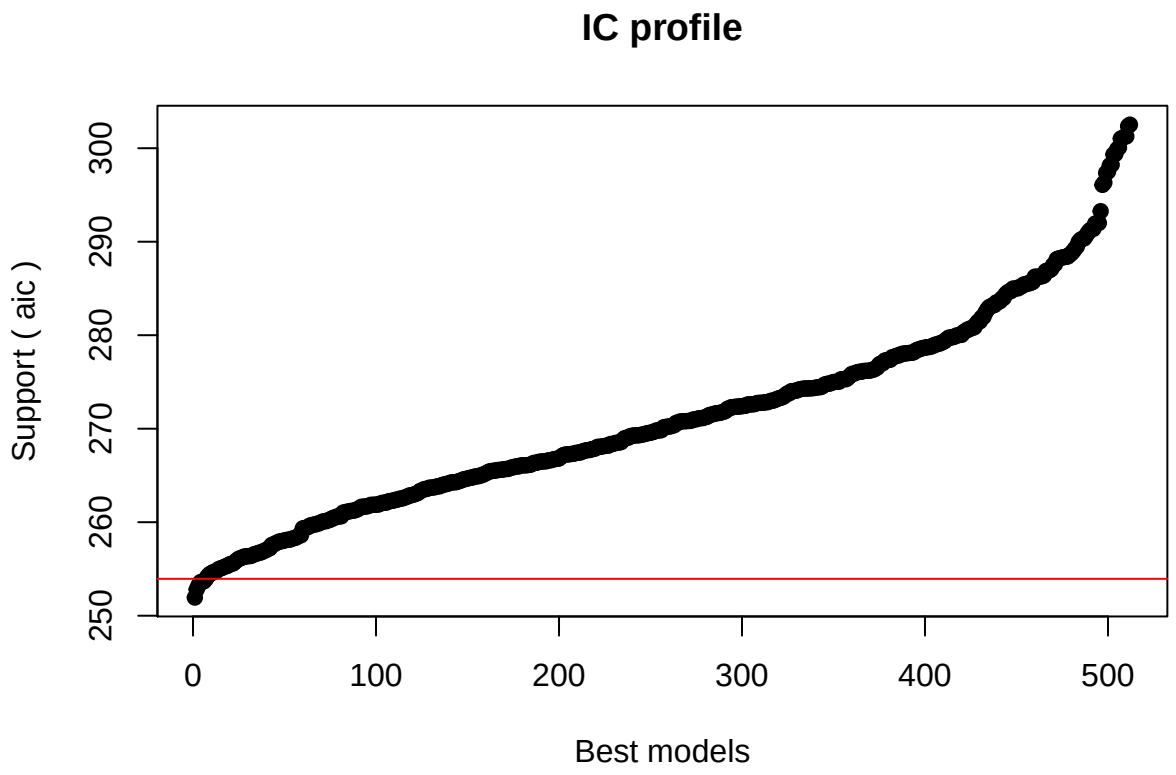
famhist is the only categorical predictor in our dataset which has the value of either 0 or 1. adiposity vs obesity, adiposity vs age are the pairs which are highly correlated predictors. We will see how our models select variables later.

2. logistic regression

All subset selection

In this section, we used glmulti package which is similar as bestglm to do all subset selection.

```
glmulti.logistic.out <-
  glmulti(chd~. , data=trainSet, level=1, fitfunction="glm", crit = "aic",
          confsetsize=512, plotty = F, report = F, family = binomial, method = "h")
plot(glmulti.logistic.out)
```




```
print(glmulti.logistic.out)
```

```
## glmulti.analysis
## Method: h / Fitting: glm / IC used: aic
## Level: 1 / Marginality: FALSE
## From 512 models:
## Best IC: 251.942240569888
## Best model:
## [1] "chd ~ 1 + tobacco + ldl + famhist + typea + age"
## Evidence weight: 0.0923754300698766
## Worst IC: 302.519026537984
## 7 models within 2 IC units.
## 72 models to reach 95% of evidence weight.
```

```
tmp <- weightable(glmulti.logistic.out)
tmp <- tmp[tmp$aic <= min(tmp$aic) + 2, ]
tmp
```

```
##                                                          model      aic    weights
## 1             chd ~ 1 + tobacco + ldl + famhist + typea + age 251.9422 0.09237543
## 2                   chd ~ 1 + tobacco + famhist + typea + age 252.8130 0.05976774
## 3   chd ~ 1 + tobacco + ldl + famhist + typea + alcohol + age 253.2404 0.04826911
## 4                       chd ~ 1 + ldl + famhist + typea + age 253.5840 0.04064906
## 5       chd ~ 1 + sbp + tobacco + ldl + famhist + typea + age 253.6491 0.03934666
## 6   chd ~ 1 + tobacco + ldl + famhist + typea + obesity + age 253.6821 0.03870279
## 7 chd ~ 1 + tobacco + ldl + adiposity + famhist + typea + age 253.9134 0.03447684
```

With level = 1, we say that we didn't consider statistical interactions first. Thus, there are $2^9 = 512$ possible models to consider here. With crit="aic", we selected the information criterion to be Akaike Information Criterion.

The plot above show the AIC values of all 512 models. The horizaontal read line differentiates between models in which AIC is less versus more than 2 units away from that of the best model with loweast AIC.

The output above shows that there are 7 models whose AIC is less than 2 units away from that of the best model. And we stored them in tmp and listed them out.

```
summary(glmulti.logistic.out@objects[[1]])
```

```
##
## Call:
## fitfunc(formula = as.formula(x), family = ..1, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8902  -0.8047  -0.4462   0.9573   2.3866
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.91292    1.38203  -5.002 5.67e-07 ***
## tobacco      0.07653    0.04083   1.874  0.06086 .
## ldl          0.15607    0.09278   1.682  0.09253 .
## famhist      1.13809    0.32087   3.547  0.00039 ***
## typea        0.04984    0.01811   2.752  0.00592 **
## age          0.04614    0.01408   3.277  0.00105 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 300.52  on 230  degrees of freedom
## Residual deviance: 239.94  on 225  degrees of freedom
## AIC: 251.94
##
## Number of Fisher Scoring iterations: 5
```

We see that the best model includes tobacco, ldl, famhist, typea and age as predictors. Highly correlated pairs adiposity vs obesity and adiposity vs age are not involved. The coeffients are reported above.

We would like to try statistical interactions and it is not hard to do that. One could do that by changing the level from 1 to 2 of glmulti(). However, it seems that even with 9 predictors, the computational workload is extremely large. I submitted it using R running on my CPU and waited for about 30 minutes but couldn't get the result.

I would do it again using GPU or HPC if possible in the future.

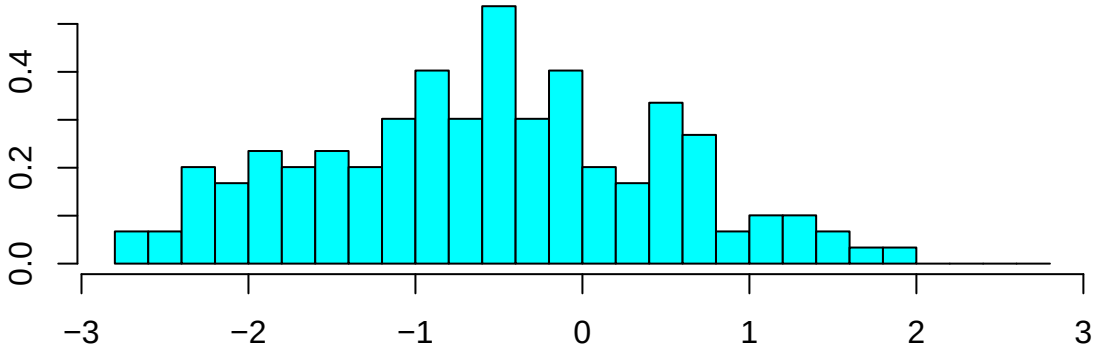
3. LDA

In this section, we used MASS package to do linear discriminant analysis.

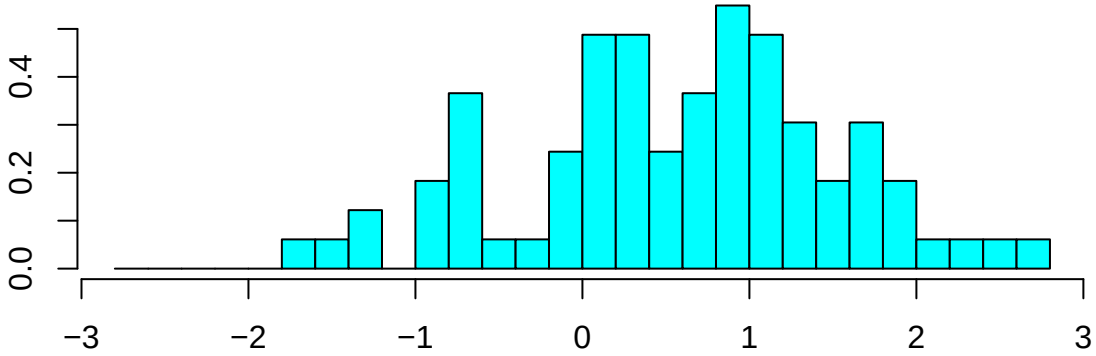
```
lda.fit <- lda(x=as.matrix(trainSet[,-10]), grouping= trainSet[,10], cv=TRUE)

lda.fit
```

```
## Call:
## lda(as.matrix(trainSet[, -10]), grouping = trainSet[, 10], cv = TRUE)
##
## Prior probabilities of groups:
##         0         1
## 0.6450216 0.3549784
##
## Group means:
##        sbp  tobacco      ldl adiposity   famhist    typea  obesity  alcohol      age
## 0 136.1208 2.577919 4.260403  23.93691 0.3087248 52.61745 25.64839 16.25732 38.72483
## 1 143.1585 5.065000 5.145732  27.52939 0.6463415 55.02439 26.42695 22.63037 49.37805
##
## Coefficients of linear discriminants:
##                    LD1
## sbp        0.004912039
## tobacco    0.068193927
## ldl        0.165025785
## adiposity  0.013254070
## famhist    1.047570466
## typea      0.037671317
## obesity   -0.051791465
## alcohol    0.003695959
## age        0.031029183
```

```
par(mar=c(5.1,4.1,0,2.1))
plot(lda.fit)
```



group 0



group 1

The LDA output indicates that estimated $\pi_1 = 0.6450$ and $\pi_2 = 0.3550$. Also, it provides group means which were used by LDA as estimates of μ_k.

The coefficients of linear discriminants output provides the linear combination of all the 9 predictors that are used to form LDA decision rule. If $\beta^T X$ is large, then the LDA classifier will predict chd = 1, and if it is small, then the LDA classifier will predict chd = 0.

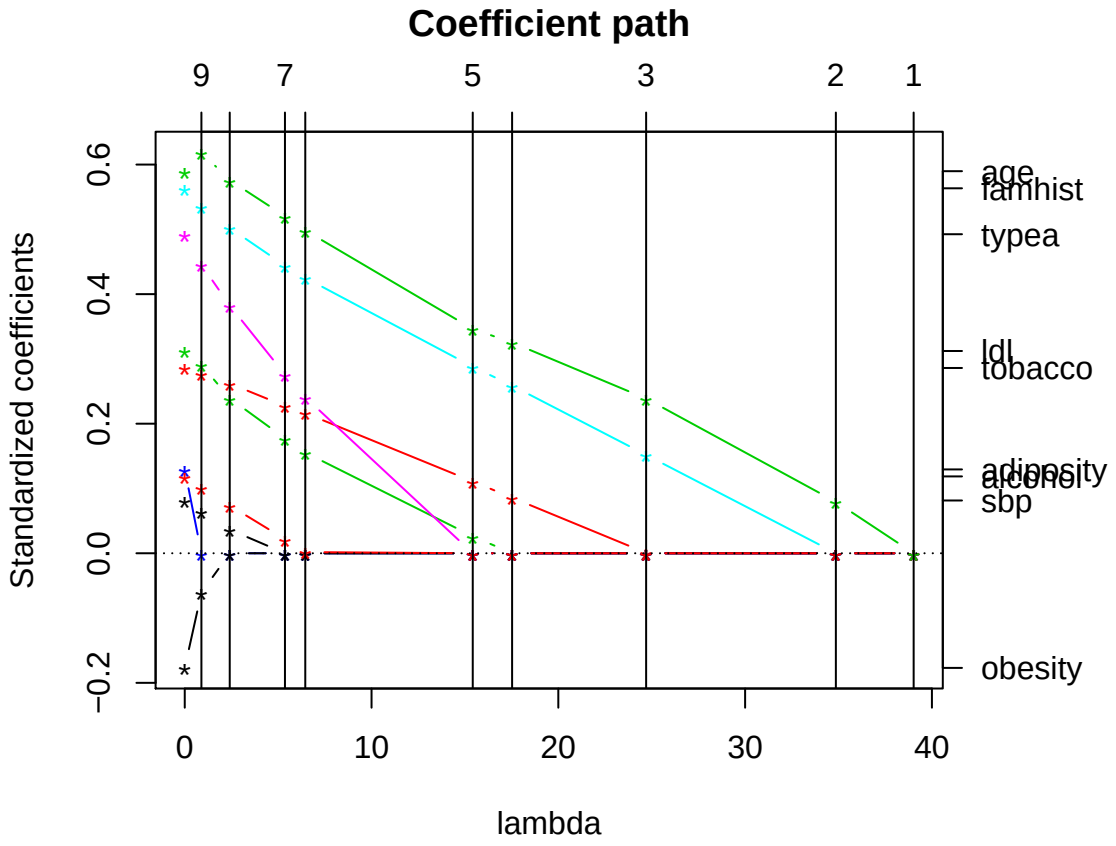
The plot() function produces plots of the linear discriminants which are approximately bell-shaped, obtained by computing $\beta^T X$ for each of the training observations.

4. logistic regression with Lasso regularization

In this section, we used glmpath package to implement lasso regularization.

Coefficients path

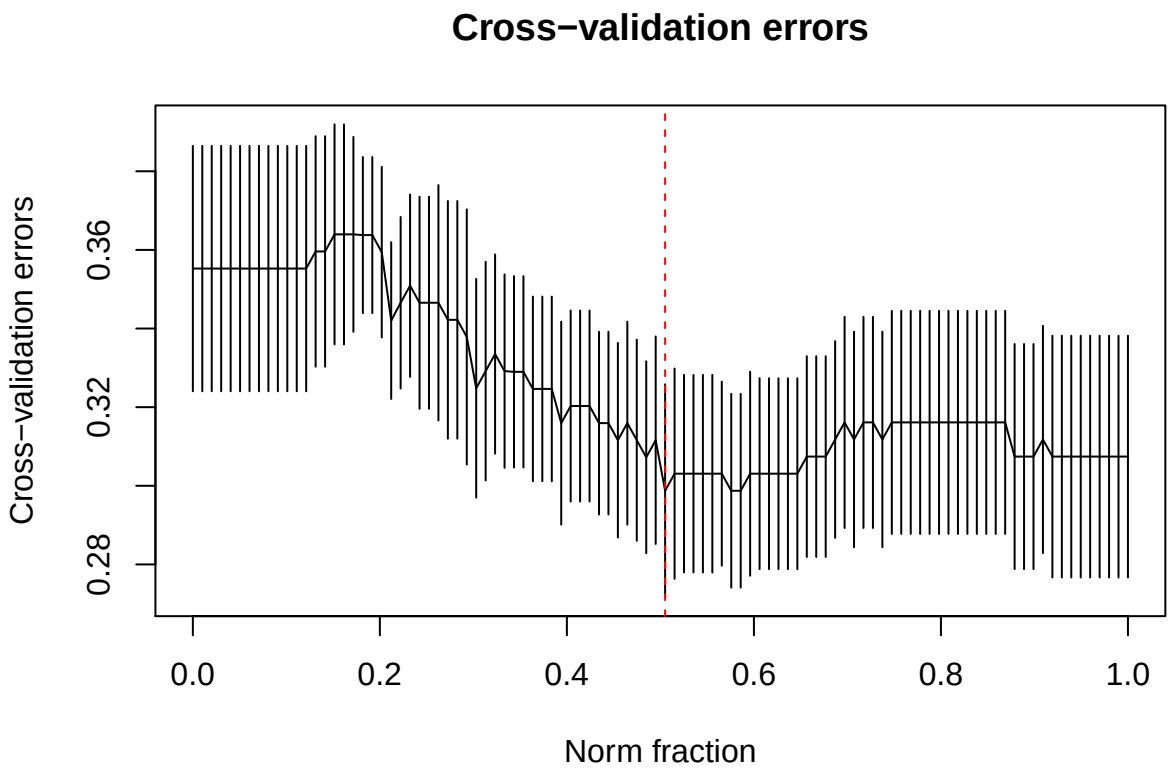
```
# fit the model with glmpath to plot the path
fit.glmpath <- glmpath(x=as.matrix(predictors[train,]),
                       y=response[train], family=binomial)

# plot the path
par(mfrow=c(1,1), mar=c(4,4,4,8))
plot(fit.glmpath, xvar="lambda")
```



From the coefficients path, we concluded that the rank of these 9 predictors regarding "importance" is age, famhist, typea, ldl, tobacco, adiposity, alcohol, sbp and obesity (from high to low).

Cross-validated predicted error plot

```
# cross-validated prediction on the training set using prediction error
fit.cv.glmpath <- cv.glmpath(x=as.matrix(predictors[train,]),
                  y=response[train],
                  family=binomial, nfold=10, plot.it=T , type="response")
cv.glmpath
cv.s <- fit.cv.glmpath$fraction[which.min(fit.cv.glmpath$cv.error)]
abline(v=cv.s, lty = 2, col = "red")
```



We used a dash red line to mark the "norm fraction of lambda" that holds minimum cross-validation errors. Note that the curve of minus log-likelihood vs norm fraction is smooth while the one of cross-validation errors vs norm fraction is rugged with ties. That's because there is no correlation between likelihood and prediction errors.

regularization parameter

```
# get the cv.s with lowest cross-validated predicted error
cv.s
```

```
## [1] 0.5050505
```

```
predict(fit.glmpath, s=cv.s, mode="norm.fraction", type="coefficients")
```

```
##                    Intercept sbp    tobacco        ldl adiposity  famhist      typea obesity
## 0.505050505050505 -3.990906   0 0.04722147 0.07330134         0 0.803498 0.01990662       0
##                         alcohol        age
## 0.505050505050505 5.833861e-05 0.03246817
## attr(,"s")
## [1] 0.5050505
## attr(,"fraction")
##         1
## 0.5050505
## attr(,"mode")
## [1] "norm.fraction"
```

For this Lasso model, we got the norm fraction of lambda 0.5050505. And the coefficients of predictors are showed. We find that sbp, adiposity and obesity are dropped because they are less "important".

Fit without lambda

```
lasso.without.fit <- glm(chd ~ 1 + tobacco + ldl + famhist + typea + alcohol +age,
                         family=binomial, data=trainSet)
summary(lasso.without.fit)
```

```
##
## Call:
## glm(formula = chd ~ 1 + tobacco + ldl + famhist + typea + alcohol +
##     age, family = binomial, data = trainSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8768  -0.8021  -0.4288   0.9661   2.3985
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.025419   1.394825  -5.037 4.73e-07 ***
## tobacco      0.070081   0.041266   1.698 0.089459 .
## ldl          0.169839   0.094230   1.802 0.071485 .
## famhist      1.125453   0.321521   3.500 0.000465 ***
## typea        0.049817   0.018142   2.746 0.006032 **
## alcohol      0.005178   0.006212   0.833 0.404599
## age          0.045610   0.014151   3.223 0.001269 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 300.52  on 230  degrees of freedom
## Residual deviance: 239.24  on 224  degrees of freedom
## AIC: 253.24
##
## Number of Fisher Scoring iterations: 5
```

Here we used the selected variables to refit the trainSet without bias. Notice that the logistic model of chd ~ 1 + tobacco + ldl + famhist + typea + alcohol +age did appear in out 7 best models in the result of glmulti. We would evaluate the performances of the two models (with and without lambda) in the section later.

5. Nearest shrunken centroids

In this section, we used pamr package to do nearest shrunken centroids.

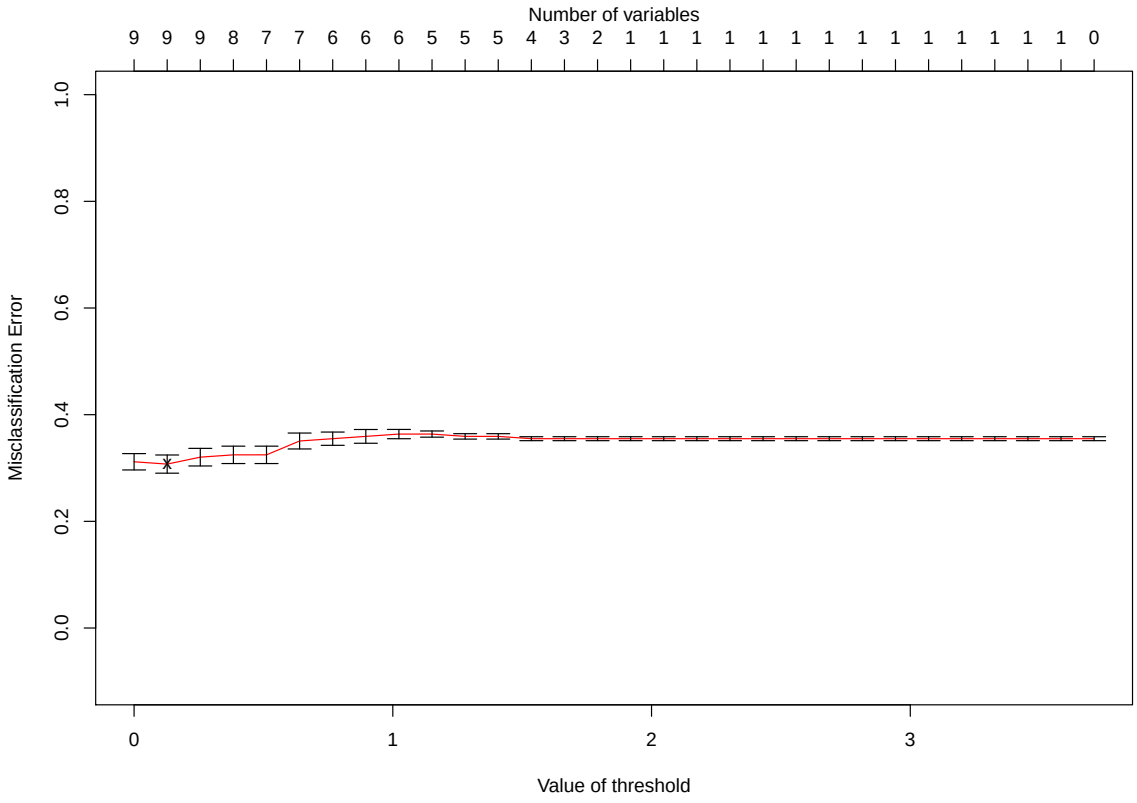
Cross validation to select threshold

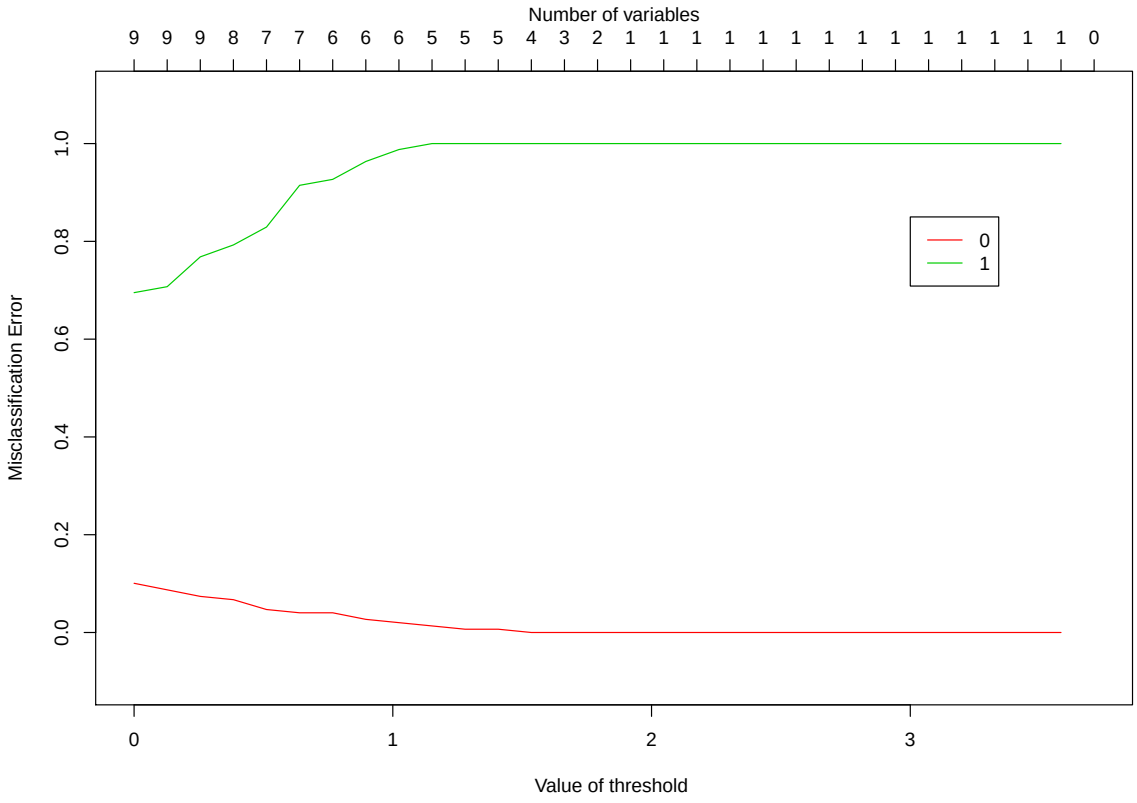
```
# Reformat the dataset for parm
pamrTrain <- list(x=t(as.matrix(trainSet[,-10])), y=trainSet[,10])
pamrValid <- list(x=t(as.matrix(testSet[,-10])), y=testSet[,10])

# Fit the classifier on the entire training set
fit.pamr <- pamr.train(pamrTrain)

# Use cross-validation to select the best regularization parameter
fit.cv.pamr <- pamr.cv(fit.pamr, pamrTrain)
```

```
fit.cv.pamr
```

```
## Call:
## pamr.cv(fit = fit.pamr, data = pamrTrain)
##    threshold nonzero errors
## 1  0.000     9       72
## 2  0.128     9       71
## 3  0.256     9       74
## 4  0.384     8       75
## 5  0.512     7       75
## 6  0.640     7       81
## 7  0.768     6       82
## 8  0.896     6       83
## 9  1.024     6       84
## 10 1.152     5       84
## 11 1.279     5       83
## 12 1.407     5       83
## 13 1.535     4       82
## 14 1.663     3       82
## 15 1.791     2       82
## 16 1.919     1       82
## 17 2.047     1       82
## 18 2.175     1       82
## 19 2.303     1       82
## 20 2.431     1       82
## 21 2.559     1       82
## 22 2.687     1       82
## 23 2.815     1       82
## 24 2.943     1       82
## 25 3.071     1       82
## 26 3.199     1       82
## 27 3.327     1       82
## 28 3.455     1       82
## 29 3.583     1       82
## 30 3.711     0       82
```

To visualize the process of cross validation, I modified pamr.plotcv() function to make the figures more elegant.


Number of variables
9 9 9 8 7 7 6 6 6 5 5 5 4 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0
Misclassification Error
0.0 0.2 0.4 0.6 0.8 1.0
0 1 2 3
Value of threshold


Number of variables
9 9 9 8 7 7 6 6 6 5 5 5 4 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0
Misclassification Error
0.0 0.2 0.4 0.6 0.8 1.0
0 1
0 1 2 3
Value of threshold

We would like to choose the best threshold to be 0.512 since it drop considerable predictors with acceptable errors.

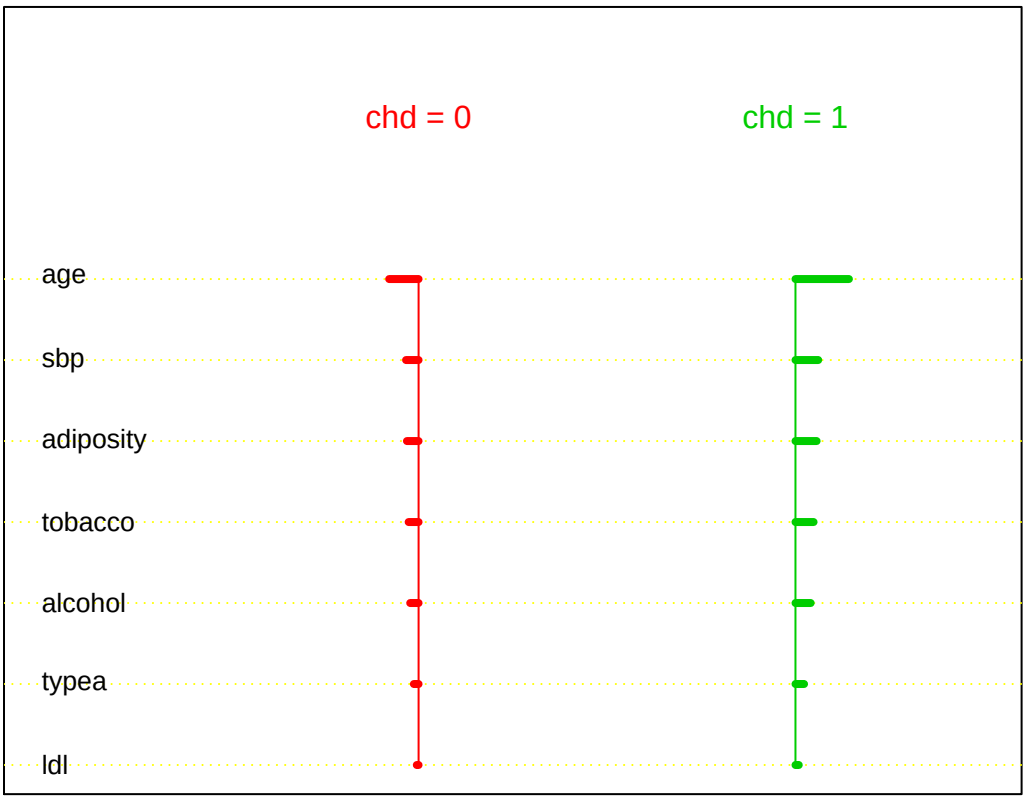
Refit the model with selected threshold

```
# Refit the classifier on the full dataset, but using the threshold
fit.pamr <- pamr.train(pamrTrain, threshold=0.512)
```

Visualize centroids of selected model

Here I defined a plot.centroids() function that is slightly different from pamr.plotcen() function because there is nothing to do with genes.

```
plot.centroids(fit.pamr, pamrTrain, 0.512)
```



Using Nearest Shrunken Centroids, we selected 7 out of 9 predictors. Notice that famhist and obesity were dropped by this method. According to domain knowledge, famihist is truely an important predictor. So eliminating famhist would lead to the low AUC value.

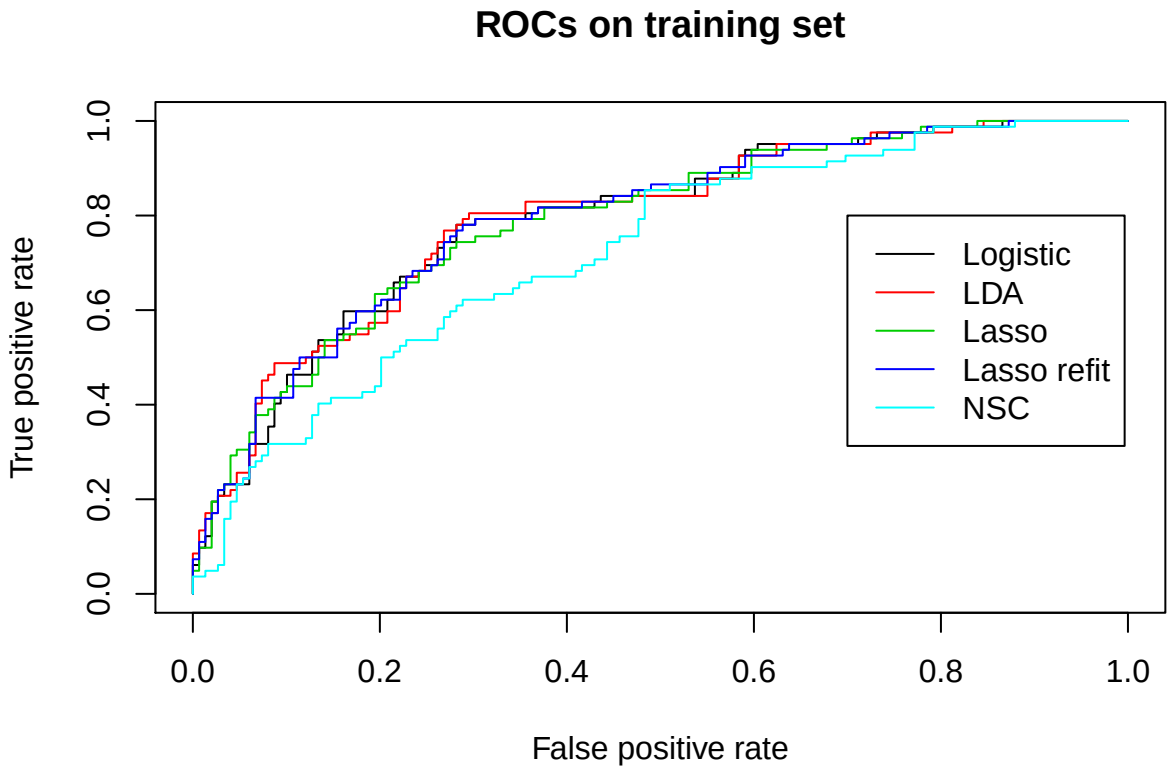
6. Evaluate the performance of the classifiers

In this section we would evaluate the performance of the classifiers using ROC curves and try to summarize our findings.

Evaluate the performance of the classifiers on the training set.

```
# logistic regression
# calculate predicted probabilities on the same training set
scores <- predict(allsubset.fit, newdata=trainSet, type="response")

# compare predicted probabilities to labels, for varying probability cutoffs
pred <- prediction(scores, labels=trainSet$chd )
perf <- performance(pred, "tpr", "fpr")

# plot the ROC curve
plot(perf, col= 1, main="ROCs on training set")
# print out the area under the curve
allsubset.train <- unlist(attributes(performance(pred, "auc"))$y.values)

# lda
# prediction on the training set
pred.lda.train <- predict(lda.fit, newx=trainSet[,-10])

# ROC on the training set
scores <- predict(lda.fit, newdata= trainSet[,-10])$posterior[,2]
pred <- prediction( scores, labels= trainSet$chd )
perf <- performance(pred, "tpr", "fpr")
plot(perf, col = 2, add=T)
# print out the area under the curve
lda.train <- unlist(attributes(performance(pred, "auc"))$y.values)

# lasso with lambda

# in-sample predictive accuracy
pred.glmpath.train <- predict(fit.glmpath, newx=as.matrix(predictors[train,]), s=cv.s,
            mode="norm.fraction", type="response")

# ROC on the training set
pred <- prediction( predictions=pred.glmpath.train, labels=response[train] )
perf <- performance(pred, "tpr", "fpr")
plot(perf, col = 3, add=T)
# print out the area under the curve
lasso.train <- unlist(attributes(performance(pred, "auc"))$y.values)

# lasso refit without lambda
# calculate predicted probabilities on the same training set
scores <- predict(lasso.without.fit, newdata=trainSet, type="response")

# compare predicted probabilities to labels, for varying probability cutoffs
pred <- prediction(scores, labels=trainSet$chd )
perf <- performance(pred, "tpr", "fpr")

# plot the ROC curve
```

```r
plot(perf, col = 4, add=T)
# print out the area under the curve
lasso.without.train <- unlist(attributes(performance(pred, "auc"))$y.values)


# Nearest Shrunken Centroids

# ROC on the training set
pred.pamr.train <- pamr.predict(fit.pamr, newx=pamrTrain$x,
                                threshold=0.512, type="posterior")[,2]
pred <- prediction(predictions=pred.pamr.train, labels= trainSet$chd)
perf <- performance(pred, "tpr", "fpr")
plot(perf, col = 5, add=T)
# print out the area under the curve
nearest.train <- unlist(attributes(performance(pred, "auc"))$y.values)

legend(0.7, 0.8, c("Logistic","LDA", "Lasso", "Lasso refit", "NSC"), lty=c(1,1), col=c(1:5))
```

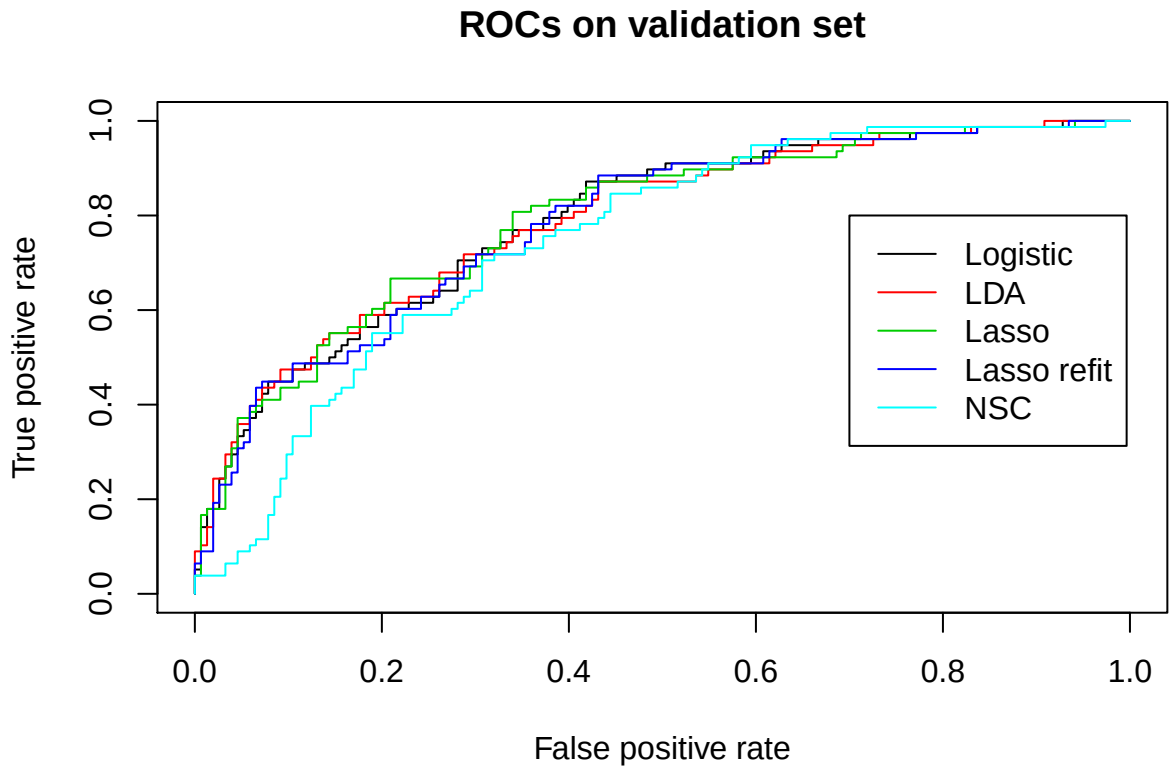


Evaluate the performance of the classifiers on the validation set.

```r
# logistic regression

# calculate predicted probabilities on validation set
scores <- predict(allsubset.fit, newdata=testSet, type="response")
```

```
# compare predicted probabilities to labels, for varying probability cutoffs
pred <- prediction(scores, labels=testSet$chd )
perf <- performance(pred, "tpr", "fpr")
# plot the ROC curve
plot(perf, col=1, main = "ROCs on validation set")
# print out the area under the curve
allsubset.test <- unlist(attributes(performance(pred, "auc"))$y.values)


# lda

# prediction on the validation set
pred.lda.test <- predict(lda.fit, newx=testSet[,-10])

# ROC on the validation set
scores <- predict(lda.fit, newdata= testSet[,-10])$posterior[,2]
pred <- prediction( scores, labels= testSet$chd )
perf <- performance(pred, "tpr", "fpr")
plot(perf, col=2, add=TRUE)
# print out the area under the curve
lda.test <- unlist(attributes(performance(pred, "auc"))$y.values)

# lasso

# predictive accuracy on a validation set
pred.glmpath.valid <- predict(fit.glmpath, newx=as.matrix(predictors[-train,]), s=cv.s,
            mode="norm.fraction", type="response")


# ROC on the validation set
pred <- prediction( predictions=pred.glmpath.valid, labels=response[-train] )
perf <- performance(pred, "tpr", "fpr")
plot(perf, col=3, add=TRUE)
# print out the area under the curve
lasso.test <- unlist(attributes(performance(pred, "auc"))$y.values)


# lasso refit

# calculate predicted probabilities on validation set
scores <- predict(lasso.without.fit, newdata=testSet, type="response")

# compare predicted probabilities to labels, for varying probability cutoffs
pred <- prediction(scores, labels=testSet$chd )
perf <- performance(pred, "tpr", "fpr")
# plot the ROC curve
plot(perf, col=4, add=TRUE)
# print out the area under the curve
lasso.without.test <- unlist(attributes(performance(pred, "auc"))$y.values)

# NSC

# ROC on the validation set
```

```
pred.pamr.valid <- pamr.predict(fit.pamr, newx=pamrValid$x,
                                threshold=0.512, type="posterior")[,2]
pred <- prediction(predictions=pred.pamr.valid, labels= testSet$chd)
perf <- performance(pred, "tpr", "fpr")
plot(perf, col=5, add=TRUE)
# print out the area under the curve
nearest.test <- unlist(attributes(performance(pred, "auc"))$y.values)

legend(0.7, 0.8, c("Logistic","LDA", "Lasso", "Lasso refit", "NSC"), lty=c(1,1), col=c(1:5))
```



Summary

Here we concluded the AUC results to get a table.

```
res <- matrix(c(allsubset.train, allsubset.test, lda.train, lda.test,
                lasso.train, lasso.test, lasso.without.train, lasso.without.test,
                nearest.train, nearest.test),
              nrow = 2, dimnames = list(c("trainingSet","validationSet"),
                                        c("Logistic", "LDA", "Lasso", "Lasso refit", "NSC")))

kable(res, title="AUCs of models")
```

	Logistic	LDA	Lasso	Lasso refit	NSC
trainingSet	0.7909642	0.7933377	0.7877721	0.7922737	0.7234408
validationSet	0.7894252	0.7888386	0.7921066	0.7847327	0.7476119

Among these 5 models, Logistic, LDA and Lasso refit have larger AUC in training set than that in validation set while Lasso and NSC have larger AUC in validation set. This is not hard to understand because it is only one experiment. If repeating several times, on average the performance on the validation set is worse than on the training set I believe.

We also find that though the AUC of validation set in NSC model is larger than that in training set, these two values are relatively small. The reason maybe that it is not the best situation to apply NSC. Nearest shrunken centroid classifier would outperform others if noise exists. Since here the number of features is relative small compared with the number of observations, it may not be a good choice to apply NSC.

The pair Lasso vs Lasso refit tells us whether to introduce bias or not. According to the AUC values, the one with bias (with lambda) outperforms. So introducing bias here seems to be a good choice to make our model more stable.

The numbers of predictors in these 5 models are 5, 9, 7, 7, 7 respectively (Logistic, LDA, Lasso, Lasso refit, NSC). So the allsubset logistic model is the most interpretable followed by Lasso, Lasso refit and NSC. LDA is the least interpretable model.

One last thing we want to mention is the set.seed() function which helps us to split the dataset. Different seed would influence the result slightly. My results above are derived from the seed 123.

7. KM problem 4.20

(a) $GaussI \leq LinLog$

These two models both have linear features. Logit function is linear regarding to X in *Linlog* while discriminant function is linear regarding to X in $GaussI$. But in LDA we maximize the joint likelihood while in Logistic regression we maximize the conditional likelihood. So *Linlog* is the one that maximize the conditional likelihood at this level.

(b) $GaussX \leq QuadLog$

The reason is similar as the one above except that Logit function and discriminant function are quadratic regarding to X. So *QuadLog* is the one that maximize the conditional likelihood at this level.

(c) $Linlog \leq QuadLog$

Generally speaking, the models with linear features belongs to the models with quadratic features. In other words, models with quadratic features form a superclass of models with linear features. So the inequation holds.

(d) $GaussI \leq QuadLog$

Combined (a) with (c), we can conclude this inequation.

(e) False

A model or a classifier that has a larger likelihood does not necessarilly mean that it would has a lower misclassification rate. One counter example can be the `minus likelihood plot` and `prediction errors plot` produced by the sample script 4_glmpath.R. We could see that the minus likelihood curve is smooth while the prediction errors curve is rugged and with ties. That implies there is probability that the misclassification error rate would remain the same even if the likelihood changes. Or more extremely, the misclassification error has the chance to increase even though the likelihood increases. Thus, we can safely draw the conclusion that the statement $L(M) > L(M')$ implies that $R(M) < R(M')$ is False.

Appendix

cv.plot function

```
cv.plot <- function(fit.cv.pamr){
  error.bars <- function(x, lower, upper, length=0.1,...){
     arrows(x,lower, x, upper, angle=90, code=3, length=length, ...)
  }
  fit <- fit.cv.pamr
  par(mar = c(5, 4.5, 5, 0.5))
  #par(mfrow = c(1, 2))
  n <- nrow(fit$yhat)
  y <- fit$y
  if (!is.null(fit$newy)) {
      y <- fit$newy[fit$sample.subset]
  }
  nc <- length(table(y))
  nfolds <- length(fit$folds)
  err <- matrix(NA, ncol = ncol(fit$yhat), nrow = nfolds)
  temp <- matrix(y, ncol = ncol(fit$yhat), nrow = n)
  ni <- rep(NA, nfolds)
  for (i in 1:nfolds) {
      ii <- fit$folds[[i]]
      ni[i] <- length(fit$folds[[i]])
      err[i, ] <- apply(temp[ii, ] != fit$yhat[ii, ], 2, sum)/ni[i]
  }
  se <- sqrt(apply(err, 2, var)/nfolds)
  plot(fit$threshold, fit$error, ylim = c(-0.1, 1),
       xlab = "Value of threshold",ylab = "Misclassification Error",
       type = "n", yaxt = "n", cex=0.6)

  axis(3, at = fit$threshold, labels = paste(fit$size), srt = 90, adj = 0)
  mtext("Number of variables", 3, 2, cex = 1)
  axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1.0))
  lines(fit$threshold, fit$error, col = 2)
  o <- fit$err == min(fit$err)
  points(fit$threshold[o], fit$error[o], pch = "x")
  error.bars(fit$threshold, fit$err - se, fit$err + se)
  err2 <- matrix(NA, nrow = length(unique(y)), ncol = length(fit$threshold))
  for (i in 1:(length(fit$threshold) - 1)) {
      s <- pamr.confusion(fit, fit$threshold[i], extra = FALSE)
      diag(s) <- 0
      err2[, i] <- apply(s, 1, sum)/table(y)
  }
  plot(fit$threshold, err2[1, ], ylim = c(-0.1, 1.1), xlab = "Value of threshold ",
      ylab = "Misclassification Error", type = "n", yaxt = "n")
  mtext("Number of variables", 3, 2, cex = 1)
  axis(3, at = fit$threshold, labels = paste(fit$size), srt = 90, adj = 0)
  axis(2, at = c(0, 0.2, 0.4, 0.6, 0.8, 1.0))
  for (i in 1:nrow(err2)) {
      lines(fit$threshold, err2[i, ], col = i + 1)
  }
  legend(3, 0.85, dimnames(table(y))[[1]], col = (2:(nc + 1)), lty = 1)
}
```

plot.centroids function

```
plot.centroids <- function (fit, data, threshold)
{
     genenames <- c("sbp", "tobacco", "ldl", "adiposity",
                    "famhist", "typea", "obesity", "alcohol", "age")
     x <- data$x[fit$gene.subset, fit$sample.subset]
     clabs <- c("chd = 0", "chd = 1")
     scen <- pamr.predict(fit, data$x, threshold = threshold,
         type = "cent")
     dif <- (scen - fit$centroid.overall)/fit$sd
     if (!is.null(fit$y)) {
         nc <- length(unique(fit$y))
     }
     if (is.null(fit$y)) {
         nc <- ncol(fit$proby)
     }
     o <- drop(abs(dif) %*% rep(1, nc)) > 0
     d <- dif[o, ]
     nd <- sum(o)
     genenames <- genenames[o]
     xx <- x[o, ]
     oo <- order(apply(abs(d), 1, max))
     d <- d[oo, ]
     genenames <- genenames[oo]
     par(mar = c(1, 5, 1, 1), col = 1)
     plot(rep(2, nd) + d[, 1], 1:nd, xlim = c(0, 2 * nc + 1),
         ylim = c(1, nd + 3), type = "n", xlab = "", ylab = "",
         axes = FALSE)
     box()
     abline(h = seq(nd), lty = 3, col = 7)
     jj <- rep(0, nd)
     for (j in 1:nc) {
         segments(jj + 2 * j, seq(nd), jj + 2 * j + d[, j], seq(nd),
           col = j + 1, lwd = 4)
         lines(c(2 * j, 2 * j), c(1, nd), col = j + 1)
         text(2 * j, nd + 2, label = clabs[j], col = j + 1)
     }
     g <- substring(genenames, 1, 20)
     text(rep(0, nd), seq(nd), label = g, cex = 0.8, adj = 0,
         col = 1)
}
```