Gulishana Adilijiang (adiliji2)

STAT 530 Bioinformatics: Homework 7

Problem1:

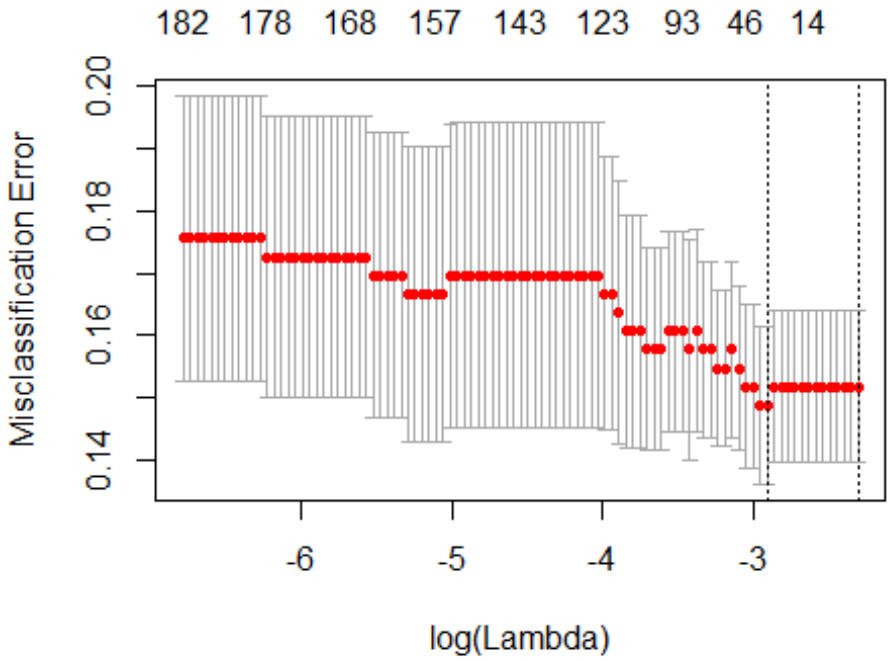
Part (a)

R Notebook

```
## Problem 1

## Preprocess the multiple myeloma data
setwd("C:/Users/Gulishana/Desktop/HW")

rm(list=ls());
library(data.table);

pData <- fread("GSE24080_series_matrix.txt",
               skip = 29, nrows = 37, header = FALSE);
expr <- fread("GSE24080_series_matrix.txt",
              skip = 67, nrow = 54675, header = TRUE, fill = TRUE);

##
Read 18.3% of 54675 rows
Read 36.6% of 54675 rows
Read 54.9% of 54675 rows
Read 73.2% of 54675 rows
Read 91.4% of 54675 rows
Read 54675 rows and 560 (of 560) columns from 0.201 GB file in 00:00:31

## training data: train = 1
train <- sapply(pData[10,-1,with=F],function(x){
  as.numeric(strsplit(x," ")[[1]][2]=="Training");
});

## remove "fake" subjects
train[1:4] = 0;

## testing data: test = 1 (categories: Training, Validation, MAQC_Remove)
test <- sapply(pData[10,-1,with=F],function(x){
  as.numeric(strsplit(x," ")[[1]][2]=="Validation");
});

## outcome
Y <- sapply(pData[15,-1,with=F],function(x){
  as.numeric(strsplit(x,": ")[[1]][3]);
});
```

```
## age
age <- sapply(pData[11,-1,with=F],function(x){
  as.numeric(strsplit(x,": ")[[1]][2]);
});

## sex
sex <- sapply(pData[12,-1,with=F],function(x){
  as.numeric(strsplit(x,": ")[[1]][2]=="female");
});

## training data
X.train <- cbind(as.matrix(t(expr[,-1,with=F]))[train==1,],
                 age[train==1],sex[train==1]);
colnames(X.train) <- c(unlist(expr[,1,with=F]),"age","sex");
Y.train <- Y[train==1];

## testing data
X.test<- cbind(as.matrix(t(expr[,-1,with=F]))[test==1,],
               age[test==1],sex[test==1]);
colnames(X.test) <- c(unlist(expr[,1,with=F]),"age","sex");
Y.test <- Y[test==1];



## Part (a)
library(glmnet)

## Warning: package 'glmnet' was built under R version 3.3.3

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.3.3

## Loaded glmnet 2.0-5

nrow(X.train)

## [1] 336

nrow(X.test)

## [1] 214



## Training: find the lambda that gives the smallest 5-fold Cross-Validation
misclassification error
CV5=cv.glmnet(X.train,Y.train,family="binomial",alpha=1,type.measure="class",
nfolds=5)
names(CV5)
```

```
##  [1] "lambda"     "cvm"        "cvsd"       "cvup"       "cvlo"
##  [6] "nzero"      "name"       "glmnet.fit" "lambda.min" "lambda.1se"

lambda=CV5$lambda.min;  lambda

## [1] 0.05414691

plot(CV5)
```




```
## Training: build a logistic regression model on the training data
fit_lasso = glmnet(X.train,Y.train,family="binomial",alpha=1,lambda=lambda)
summary(fit_lasso)

##            Length Class     Mode
## a0             1  -none-    numeric
## beta       54677  dgCMatrix S4
## df             1  -none-    numeric
## dim            2  -none-    numeric
## lambda         1  -none-    numeric
## dev.ratio      1  -none-    numeric
## nulldev        1  -none-    numeric
## npasses        1  -none-    numeric
## jerr           1  -none-    numeric
## offset         1  -none-    logical
## classnames     2  -none-    character
```

```
## call           6  -none-    call
## nobs           1  -none-    numeric
```

```
head(fit_lasso$beta)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##           s0
## 1007_s_at  .
## 1053_at    .
## 117_at     .
## 121_at     .
## 1255_g_at  .
## 1294_at    .
```

```
which(fit_lasso$beta!=0)
```

```
##  [1]   247  3447  6349  8330  8786  9387 10469 11039 17300 17969 19162
## [12] 19826 26984 27187 33776 34879 35808 36430 36929 38264 41304 42488
## [23] 44136 48903 50894 53055 54205 54479
```

```
length(which(fit_lasso$beta!=0))
```

```
## [1] 28
```

```
## Test: use cutoff c=0.5 to report misclassification rate of the test data


pred.Y.test=predict.glmnet(fit_lasso,X.test,type="response")

misclass_rate= mean( ifelse( Y.test!=(pred.Y.test>=0.5),1,0 ) ); misclass_rat
e
```

```
## [1] 0.1261682
```

Result:

There are **28** non-zero estimated coefficients of the model built on the training data.

And the misclassification rate of the test data is **12.61682%**.

In fact, each time I ran the model, it would end up with a different value of lambda that gave the smallest 5-fold cross-validation misclassification error, as well as a different number of non-zero estimated coefficients of the model.

However, the misclassification rate of the test data did not change for different models, which is consistently **12.61682%**.

Part (b)

R Notebook

```
## Part (b)

## Calculate the ratio of "0" and "1" in the true response of the test data, respectively


length(which(Y.test==1))/length(Y.test)

## [1] 0.1261682

length(which(Y.test==0))/length(Y.test)

## [1] 0.8738318
```

Result:

The binary outcome in the test data, "Y.test", contains **12.61682%** of values as "1", and **87.38318%** of values as "0". Since value "0" is the majority in this case, based on the way a logistic regression model uses to predict test responses, the classifier would basically try to correctly predict and distinguish the "1"s from the "0"s in the test responses.

Therefore, the dumbest classifier would misclassify all the minority-value "1" into the class of "0", which gives the misclassification rate that equals to the proportion of value "1" in the "Y.test", that is **12.61682%**. In this case, the lasso classifier has not done a good job for yielding a consistent misclassification rate of **12.61682%**.

However, an extremely dumber logistic regression classifier may misclassify all the majority-value "0" into the class of "1". I don't know if this is possible, if yes, it will yield a misclassification rate of **87.38318%**. In this case, the lasso classifier has done a good job.

Part (c)

R Notebook

```
## Part (c)

## Preprocess data
ncol(X.train)

## [1] 54677

ncol(X.test)

## [1] 54677

X.train=X.train[,1:2000]
X.test=X.test[,1:2000]
data_training=data.frame(Y=Y.train,X=X.train)
data_test=data.frame(Y=Y.test,X=X.test)



## Training: build a random forest model on the training data
library(ranger)

## Warning: package 'ranger' was built under R version 3.3.3

fit_rf=ranger(Y~.,data=data_training,classification=TRUE,write.forest = TRUE)
summary(fit_rf)

##                             Length Class         Mode
## predictions                 336    -none-        numeric
## num.trees                     1    -none-        numeric
## num.independent.variables     1    -none-        numeric
## mtry                          1    -none-        numeric
## min.node.size                 1    -none-        numeric
## prediction.error              1    -none-        numeric
## forest                        9    ranger.forest list
## treetype                      1    -none-        character
## call                          5    -none-        call
## importance.mode               1    -none-        character
## num.samples                   1    -none-        numeric



## Test: use cutoff c=0.5 to report misclassification rate of the test data

pred.Y.test=predict(fit_rf,data=data_test,type="response")
pred.Y.test=pred.Y.test$predictions

misclass_rate= mean( ifelse( Y.test!=(pred.Y.test>=0.5),1,0 ) ); misclass_rat
e

## [1] 0.135514
```