Classification in the BreastCancer data

Jack Young

2022-11-01

Cleaning the data

Before we start drawing any insights from the data, we want to first clean it up into a workable form. Firstly, note that there are some missing observations on predictors in the data, which have been encoded as `NA`. Thankfully, there are very little of these observations relative to the total number of observations in the data, so we can simply omit these. Additionally, we have that the nine cytological characteristics in the data are ordinal variables on a 1-10 scale, encoded as factors in the `BreastCancer` data. For our analysis, we can convert these factor variables into numerical variables - we can assume this is a reasonable approach to take as we can more or less assume equal distancing between the 10 levels, and the 1-10 ordering is preserved resulting in no loss of information. However, we also then standardise each of the 9 cytological characteristics, as this accounts for differences in variability across the data.

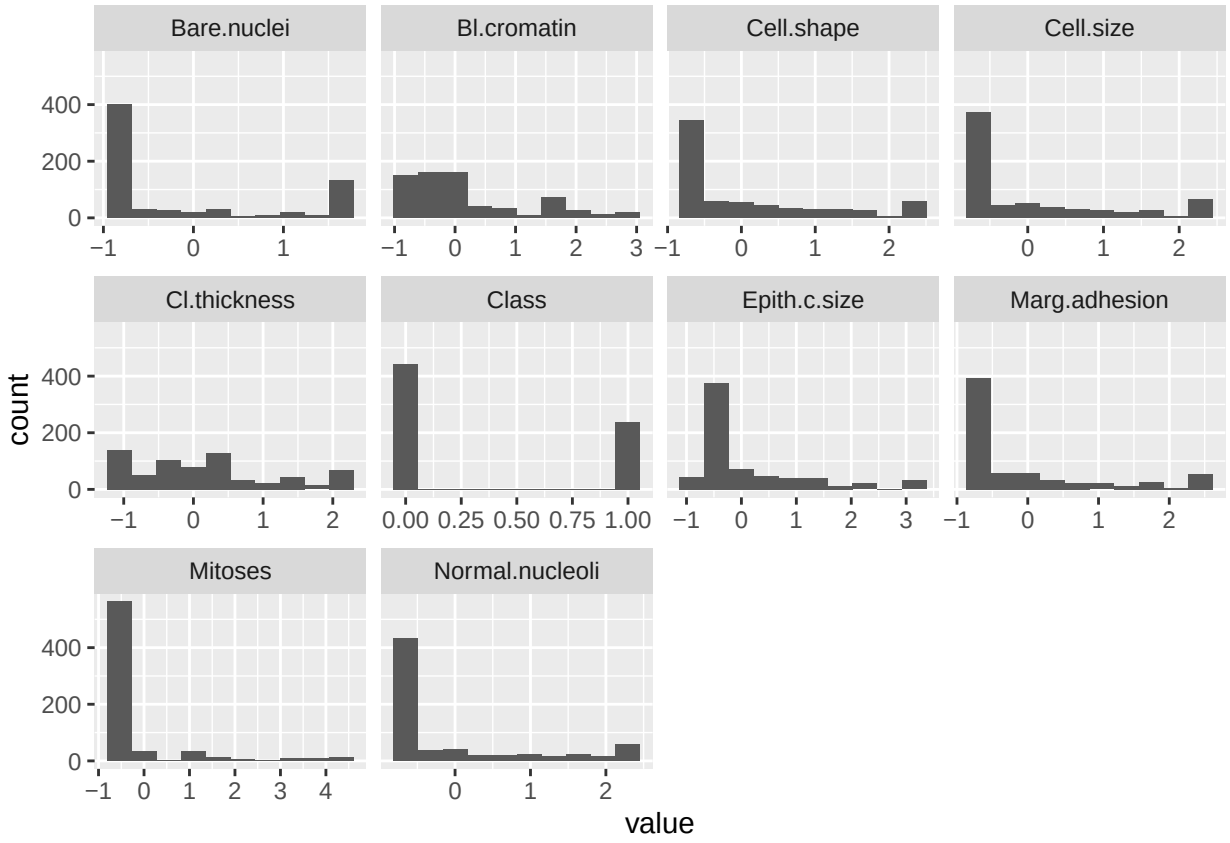
Exploring the data

Now our data is in the correct format, we begin our exploration - a good place to start is the `summary` function:

```
##       Id             Cl.thickness       Cell.size         Cell.shape
##  Length:683         Min.   :-1.2203   Min.   :-0.7017   Min.   :-0.7412
##  Class :character   1st Qu.:-0.8658   1st Qu.:-0.7017   1st Qu.:-0.7412
##  Mode  :character   Median :-0.1568   Median :-0.7017   Median :-0.7412
##                     Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##                     3rd Qu.: 0.5523   3rd Qu.: 0.6033   3rd Qu.: 0.5972
##                     Max.   : 1.9703   Max.   : 2.2345   Max.   : 2.2702
##  Marg.adhesion      Epith.c.size      Bare.nuclei       Bl.cromatin
##  Min.   :-0.6389   Min.   :-1.0050   Min.   :-0.6983   Min.   :-0.9981
##  1st Qu.:-0.6389   1st Qu.:-0.5552   1st Qu.:-0.6983   1st Qu.:-0.5899
##  Median :-0.6389   Median :-0.5552   Median :-0.6983   Median :-0.1817
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4084   3rd Qu.: 0.3444   3rd Qu.: 0.6738   3rd Qu.: 0.6347
##  Max.   : 2.5029   Max.   : 3.0434   Max.   : 1.7716   Max.   : 2.6758
##  Normal.nucleoli      Mitoses            Class
##  Min.   :-0.6125   Min.   :-0.3561   Min.   :0.0000
##  1st Qu.:-0.6125   1st Qu.:-0.3561   1st Qu.:0.0000
##  Median :-0.6125   Median :-0.3561   Median :0.0000
##  Mean   : 0.0000   Mean   : 0.0000   Mean   :0.3499
##  3rd Qu.: 0.3703   3rd Qu.:-0.3561   3rd Qu.:1.0000
##  Max.   : 2.3358   Max.   : 4.5329   Max.   :1.0000
```

We can already start to draw some basic insights from the data - for example, notice that all of the nine cytological characteristics in the data have mean values 0, as they have been standardised (they will also

have standard deviation = 1). Notice also, that our `Class` variable has a mean value of around 0.35 - in the context of our investigation, this means that around 35% of the observations we are considering in our analysis were malignant, and the remaining were benign. To further grasp the distribution of values, we can plot histograms:

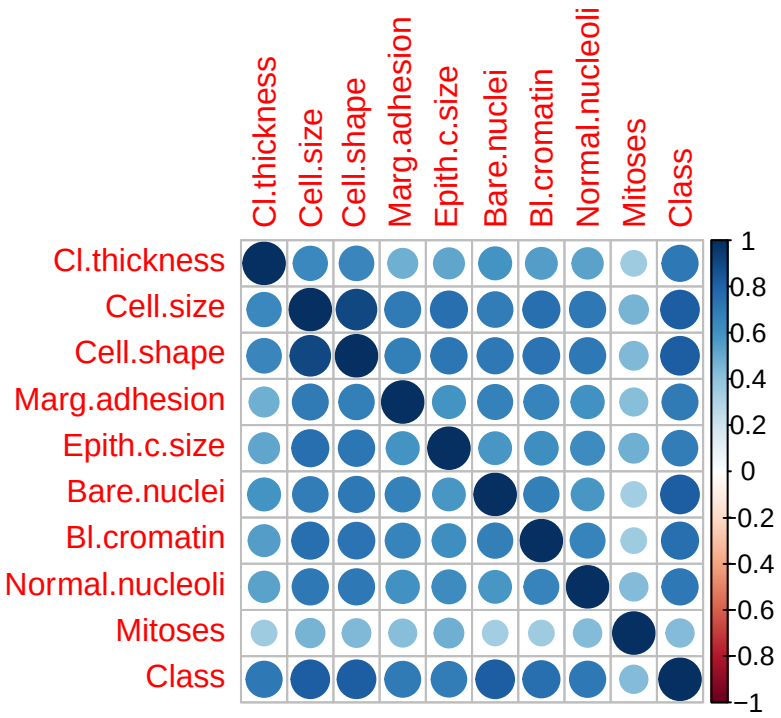


Taking into account the fact that the cytological characteristics were originally measured on a 1-10 scale before standardising, we notice that the majority of our nine cytological characteristics have lower values on the 1-10 scale on average, but now we also notice that for a number of these variables, such as `Mitoses` and `Cell.size`, the majority have been given the lowest possible score of 1. Others exhibit a slightly different distribution however, such as `Cl.thickness` which has a more even distribution of ordinal scores.

Now, we consider the relationship between variables in our data. Before we do this, consider the dimensions of our data:

```
## [1] 683  11
```

This means we have 683 observations on 11 variables. One option would be to produce a scatterplot matrix on our data, but this is unlikely to be effective given its high dimensionality. Alternatively, we can consider the correlation matrix to examine the relationship between variables - which is best visualized using a heatmap which colour codes pairs of variables based on the value of their Pearson correlation coefficient. Omitting the `ID` column, we do this like so:



We see instantly that positive correlation is exhibited between all pairs of numeric variables in the data. We note that the `Mitoses` variable appears to have the lowest correlation values across the board, and the largest actually was our `Class` variable - so it would initially appear to be the case that the higher our nine explanatory variables were ranked on our 1-10 scale, the higher chance of that particular cell being malignant.

To further explore this idea, we split our data into benign and malignant observations, and take the column means of each variable, allowing us to easily compare the two subsets:

```
##           Cl.thickness  Cell.size Cell.shape Marg.adhesion Epith.c.size
## Benign      -0.5240440 -0.6017657 -0.6025644    -0.5178153   -0.5065718
## Malignant    0.9735377  1.1179245  1.1194084     0.9619665    0.9410791
##           Bare.nuclei Bl.cromatin Normal.nucleoli    Mitoses
## Benign     -0.6031546   -0.555890      -0.5268939 -0.3162029
## Malignant   1.1205047    1.032699       0.9788322  0.5874230
```

Providing some confirmation to the findings from our correlation heatmap, we see that malignant observations exhibited higher values across the board for our explanatory variables.

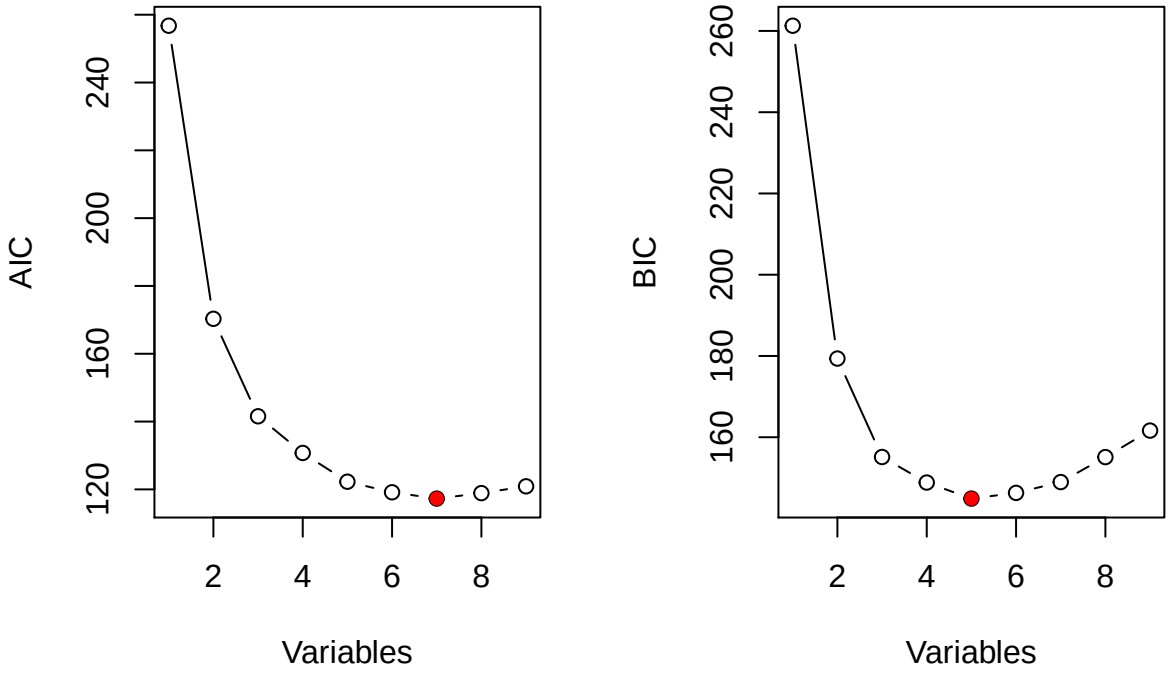
Classifying the data

Logistic regression

The first method of classification we consider is logistic regression - as our outcome (Class) is a binary variable, we can build a model to predict this based on the independent variables in our data. However with logistic regression, we only want to include explanatory variables that have a statistically significant relationship with the response variable, in order to avoid the issue of multicollinearity. In this investigation, we do this through subset selection - more specifically, the best subset selection algorithm. This works by

using least squares, fitting a regression model for each possible subset of our explanatory variables. That is, for $k = 1, 2, ...p$, the model $\mathcal{M}_k$ is fitted, that is, the model with a subset of k explanatory variables that has the smallest residual sum of squares (or R^2). After having done so, we select the "best" model among these - however the model you select may depend on the statistic you are considering, as we can use either the AIC or BIC which draw differing conclusions, as shown by the graphs below:

```
## Morgan-Tatar search since family is non-gaussian.
## Morgan-Tatar search since family is non-gaussian.
```



In order to remove some of this ambiguity, we shall use cross validation, noting that we have already used the `set.seed()` function so that the analysis is reproducible. We consider 10-fold cross validation, which will split our data into 10 folds of roughly the same size and assign our observations to one of these folds. For fold k, we first fit the model to the training data (i.e. all data that falls outside fold k), and then use this model to predict values of the response variable in the training set, from which we can calculate the MSE. In the context of the best subset selection we have used, we shall use our 10-fold cross validation on each of the $\mathcal{M}_k$ we found earlier, and find the MSE of each of these models. We can then choose an M_k based on that which minimises the MSE.

```
## [1] "The CV error obtained from the optimal model according to AIC BSS is: 0.0386114"
```

```
## [1] "The SV error obtained from the optimal model according to BIC BSS is: 0.0410512"
```

```
##
## Call:  glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
```

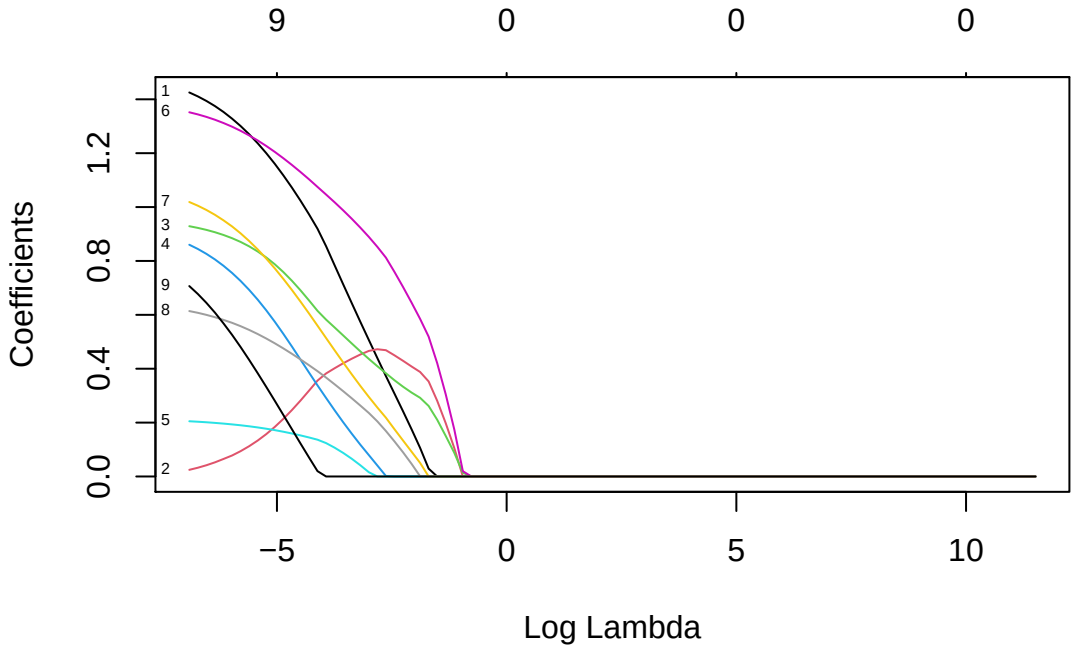
```
## Coefficients:
##     (Intercept)     Cl.thickness       Cell.shape    Marg.adhesion
##         -1.0722           1.5070           1.0311           0.9814
##     Bare.nuclei      Bl.cromatin  Normal.nucleoli          Mitoses
##          1.4149           1.1323           0.6905           0.8760
##
## Degrees of Freedom: 682 Total (i.e. Null);  675 Residual
## Null Deviance:       884.4
## Residual Deviance: 103.3     AIC: 119.3
```

We therefore can build our chosen M_k, and extract the coefficients from the above summary. We notice that (aside from the intercept, of course) all of the coefficients in this model are positive-valued, the greatest of which is `Cl.thickness`, suggesting the higher this value is on our scale, the greater the likelihood of the observation being malignant.

Regularized logistic regression

We now attempt to build a classifier based on a regularized form of logistic regression - these methods shrink the coefficient estimates in regression by smoothing the least squares loss function, and in our analysis we shall consider the LASSO method. The method works such that as λ increases, the closer the regression coefficient estimates are to 0, and the smaller λ is, the closer the coefficient estimates are to the least squares estimates.

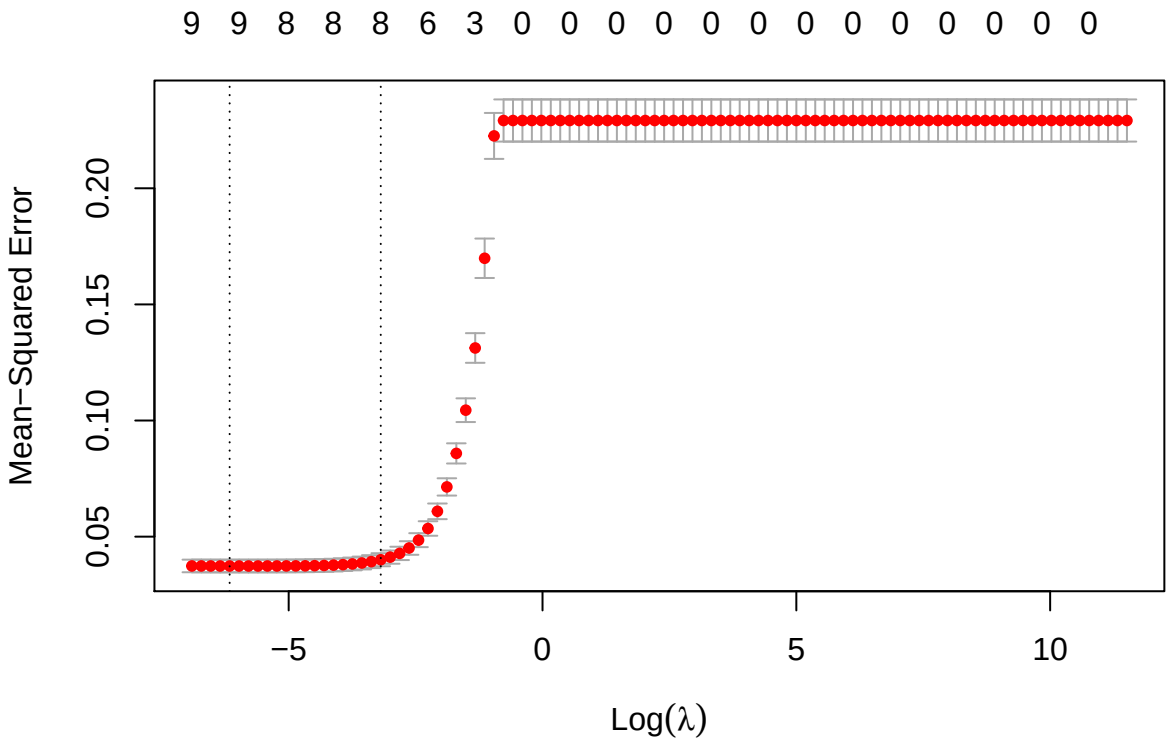
Now we have fit our GLM for the specified grid of λ values, we can examine how varying our value of (logged) λ affects the shrinking of the coefficients in our model - we can visualize this through the following plot:



Now using 10-fold cross validation once again, we can plot how varying our value of λ affects the estimated MSE of the method:

```
## Warning: Only mse, deviance, mae available as type.measure for Gaussian models;
```

```
## mse used instead
```



Furthermore - we can extract the optimal value for λ and the associated MSE estimate:

```
## [1] "Optimal tuning parameter value: 0.0021049"
```

```
## [1] "Optimal tuning parameter MSE: 0.03735276"
```

Using this value of λ, we can check the coefficients of the model that results:

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)     -1.09969558
## Cl.thickness     1.35309760
## Cell.size        0.06495791
## Cell.shape       0.89646461
## Marg.adhesion    0.78208300
## Epith.c.size     0.19625418
## Bare.nuclei      1.31201518
## Bl.cromatin      0.95082523
## Normal.nucleoli  0.58126361
## Mitoses          0.56838304
```

The model presented above is different from the model found as a result of best subset selection, in the sense that this model has preserved each of the nine explanatory variables. There are however, some similarities. For example, the intercept has once again been attributed a negative value of similar magnitude, and all explanatory variables are positively-valued with `Cl.thickness` the greatest of the set.

Discriminant Analysis

The final method we consider in our analyses is discriminant analysis, both QDA and LDA. To carry out this section, we can use both the `linDA` and `quaDA` functions, but we note that this time we are asked to consider every single subset of explanatory variables in the data, and select a subset of variables based on that which minimises the cross-validation estimate of the test error. We can carry out 10-fold cross-validation using the `validation` argument that is passed to both `linDA` and `quaDA`, and compare the best models for each of the methods, not only in terms of their MSE estimate but also the confusion matrices which show the nature of the misclassifications that each of the methods tend to make:

```
## [1] "LDA error rate:0.03953148"
```

```
##          predicted
## original   0   1
##        0 437   7
##        1  20 219
```

```
## [1] "QDA error rate:0.04245974"
```

```
##          predicted
## original   0   1
##        0 427  17
##        1   6 233
```

We notice that LDA exhibits a lower error rate than QDA in this example, but notice the difference exhibited by the confusion matrices - the errors made by LDA tend to misclassify malignant cells to benign more often, whereas QDA tends to more frequently misclassify benign cells to malignant. This could be a shortcoming of LDA in a real-world context, as it would appear it misses malignant cells more frequently.

Choosing our model

Now we have considered all of the above models, we must come to a conclusion as to which is the best performing model of the options available. As we have used 10-fold cross validation to test the accuracy of each of the models, we have a statistic that is a fair measure to use for comparison. In our testing, the best performing of the models was LASSO, with an error rate of 0.03735, taken to 4 significant figures.

Code appendix

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(1)
library(bestglm)
library(glmnet)
library(nclSLR)
library(tidyverse)
library(mlbench)
library(ggplot2)
library(leaps)
data(BreastCancer)
```

```r
#### DATA PRE-PROCESSING, EXPLORATORY ANALYSIS

## Remove NA observations from the data
fully_observed_bc_data <- na.omit(BreastCancer)
## Convert factors to numeric
num_dat <- mutate_if(fully_observed_bc_data, is.factor, ~as.numeric(.x))
## Code our Class variable as 1s and 0s - where 0 represents a benign cell
## and 1 represents a malignant cell
num_dat$Class <- num_dat$Class - 1
dat <- cbind.data.frame(num_dat$Id, scale(num_dat[,2:10]), num_dat$Class)
names(dat)[1] <- "Id"
names(dat)[11] <- "Class"

## Summarise the data
summary(dat)

## Plot histograms of the numeric variables
ggplot(gather(dat[,2:11]), aes(value)) +
  geom_histogram(bins = 10) +
  facet_wrap(~key, scales = 'free_x')

## Check the dimensions of the data
dim(dat)

## Take the correlation matrix of all numeric variables in the data
cor_matrix <- cor(dat[,2:11])
corrplot::corrplot(cor_matrix)

## Form subsets of the data according to their class
benign_dat <- subset(dat, Class==0)
malignant_dat <- subset(dat, Class==1)

## Take the column means of the numeric values in each subset
means <- rbind(colMeans(benign_dat[,2:10]), colMeans(malignant_dat[,2:10]))
rownames(means) <- c("Benign", "Malignant")
print(means)
#bss_fit <- glm(Class ~.,data = dat[,2:11], family = binomial)

aic_bestglm <- bestglm(dat[,2:11], family = binomial, IC="AIC", method = "exhaustive")
bic_bestglm <- bestglm(dat[,2:11], family = binomial, IC="BIC", method = "exhaustive")
best_bic = which.min(bic_bestglm$Subsets$BIC) -1
best_aic = which.min(aic_bestglm$Subsets$AIC) -1
p = 9
## Create multi-panel plotting device
par(mfrow=c(1,2))
plot(aic_bestglm$Subsets$AIC[2:10], type = 'b', ylab = "AIC", xlab = 'Variables')
points(best_aic, aic_bestglm$Subsets$AIC[best_aic+1], col="red", pch=16)

plot(bic_bestglm$Subsets$BIC[2:10], type = 'b', ylab = "BIC", xlab = 'Variables')
points(best_bic, bic_bestglm$Subsets$BIC[best_bic+1], col="red", pch=16)
## We define a function to estimate the average mean squared error
## using general K-fold cross validation
reg_cv = function(X1, y, fold_ind) {
```

```
  Xy = data.frame(X1, y=y)
  nfolds = max(fold_ind)
  if(!all.equal(sort(unique(fold_ind)), 1:nfolds)) stop("Invalid fold partition.")
  cv_errors = numeric(nfolds)
  for(fold in 1:nfolds) {
    tmp_fit = lm(y ~ ., data=Xy[fold_ind!=fold,])
    yhat = predict(tmp_fit, Xy[fold_ind==fold,])
    yobs = y[fold_ind==fold]
    cv_errors[fold] = mean((yobs - yhat)^2)
  }
  fold_sizes = numeric(nfolds)
  for(fold in 1:nfolds) fold_sizes[fold] = length(which(fold_ind==fold))
  test_error = weighted.mean(cv_errors, w=fold_sizes)
  return(test_error)
}

## Use the reg_cv function to estimate the avg mean square error across all
## of the best possible models using each number of explanatory variables
reg_bss_cv = function(X1, y, best_models, fold_index) {
  p = ncol(X1)
  test_errors = numeric(p)
  for(k in 1:p) {
    test_errors[k] = reg_cv(X1[,best_models[k,]], y, fold_index)
  }
  return(test_errors)
}

## We use 10-sample cross validation
fold_index <- sample(1:10, nrow(dat), replace = TRUE)

## Use our functions to estimate the MSE for each of the best models
bss_mse_aic = reg_bss_cv(dat[,2:10], dat$Class, as.matrix(aic_bestglm$Subsets[2:10, 2:10]), fold_index)

bss_mse_bic = reg_bss_cv(dat[,2:10], dat$Class, as.matrix(bic_bestglm$Subsets[2:10,2:10]), fold_index)
## Consider the number of variables in the model that minimises the
## estimated MSE
cv_aic <- bss_mse_aic[best_aic]
cv_bic <- bss_mse_bic[best_bic]

print(paste0("The CV error obtained from the optimal model according to AIC BSS is: ", round(cv_aic, 8)))
print(paste0("The SV error obtained from the optimal model according to BIC BSS is: ", round(cv_bic, 8)))


aic_bestglm$BestModel


#### LASSO CODE

## Choose grid of values for tuning parameter
grid = 10^seq(5, -3, length=100)
## Fit a LASSO regression model for each value of the tuning parameter
lasso_fit = glmnet(dat[,2:10], dat$Class, alpha=1, standardize = FALSE, lambda = grid, family = "binomia
```

```r
beta1_hat = coef(lasso_fit)
plot(lasso_fit, xvar = "lambda", col=1:9, label=TRUE)

lasso_cv_fit = cv.glmnet(as.matrix(dat[,2:10]), dat$Class, alpha = 1, standardize = FALSE, lambda = gri
plot(lasso_cv_fit)

## Extract optimal tuning parameter value
lambda_min = lasso_cv_fit$lambda.min
## Extract index of optimal tuning parameter
i = which(lasso_cv_fit$lambda == lasso_cv_fit$lambda.min)


print(paste0("Optimal tuning parameter value: ", round(lambda_min,8)))
print(paste0("Optimal tuning parameter MSE: ", round(lasso_cv_fit$cvm[i], 8)))


coef(lasso_fit, s = lambda_min)


#### DISCRIMINANT ANALYSIS CODE

## Calculate all subsets to index our explanatory variables
all_subsets <- lapply(1:9, combn, x = c(2:10), simplify = FALSE)
##Set initial best error rate
best_error_rate_lda <- 1
best_error_rate_qda <- 1

## Loop through all subsets of explanatory variables
for (i in 1:9){
  num_of_size_i <- length(all_subsets[i][[1]])
  for (j in 1:num_of_size_i){
    ## Perform lda for each possible subset
    lda_fit = linDA(variables = as.matrix(dat[,all_subsets[i][[1]][j][[1]]]), group = dat$Class, valida
    current_error_rate <- lda_fit$error_rate
    ## Store the current model as the best model if it beats the
    ## previous best error rate
    if (current_error_rate < best_error_rate_lda){
      best_error_rate_lda <- current_error_rate
      best_lda_fit <- lda_fit
    }
  }
}

## Loop through all subsets of explanatory variables
for (i in 1:9){
  num_of_size_i <- length(all_subsets[i][[1]])
  for (j in 1:num_of_size_i){
    ## Perform Qda for each possible subset
    qda_fit = quaDA(variables = as.matrix(dat[,all_subsets[i][[1]][j][[1]]]), group = dat$Class, valida
    current_error_rate <- qda_fit$error_rate
    ## Store the current model as the best model if it beats the
    ## previous best error rate
    if (current_error_rate < best_error_rate_qda){
```

```
      best_error_rate_qda <- current_error_rate
      best_qda_fit <- qda_fit
    }
  }
}

print(paste0("LDA error rate:" ,round(best_error_rate_lda,8)))
print(best_lda_fit$confusion)
print(paste0("QDA error rate:" ,round(best_error_rate_qda,8)))
print(best_qda_fit$confusion)
```