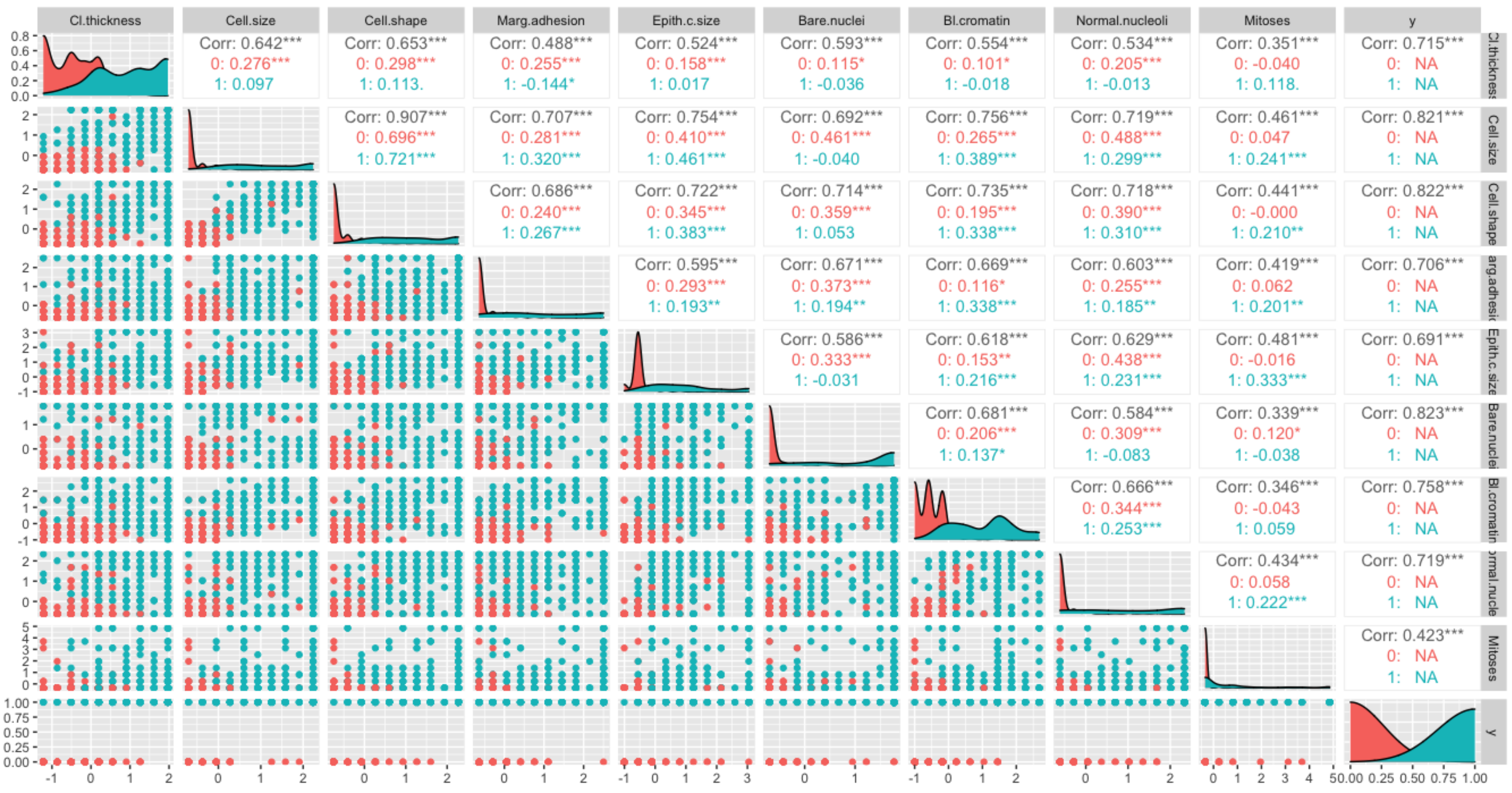
Introduction:

The data set for this project includes characteristics of breast tissue samples that determine if the tissue sample can be classified as benign or malignant.
The data set consists of 9 categorical explanatory variables that are used in the classification of the tissues. The response variable is the class variable, which is binary that takes either the value "benign" or "malignant". Here it is important to note that the data set includes an "Id" column which is the unique id for each patient in the data set. This variable is not important for the analysis, so it is removed from the data set. Moreover the data consist of 699 patients. However, for some patients, specific values for some of the explanatory variables are missing. These patients are removed for the data set in order to proceed with the analysis. Out of the 699 patients, 16 patients have missing values, so the final dataset ,that is used for building a classifier for the "Class" variable, consists of 683 patients. Note also that the predictor variables, as well as the response one, are factors. In order to proceed with the analysis and apply the appropriate statistical concepts on the data set, these factors are converted to quantitative variables (appendix 1.1). The response variable is transformed into a vector that contains 0s a 1s, where 0 represents "benign" and 1 "malignant". Finally, the predictor variables are standardised prior to the analysis (appendix 1.2).
Moving on to the data set, out of the 683 tissue samples, 65% (444) of them were classified as benign and 35% (239) as malignant. Also the mean values (appendix 1.2) of the predictor variables for the tissues that were classified as malignant appear to be higher compared to the sample means when the tissues were classified as benign. These differences can be found in the table below:

Class	Cell Thickness	Cell Size	Cell Shape	Marg. adhesion	Epith. C size	Bare. nuclei	Bl. cromatin	Normal nucleoli	Mitoses
Benign	-0.524	- 0.602	-0.603	-0.518	-0.507	-0.603	-0.556	-0.527	-0.310
Malignant	0.974	1.118	1.119	0.962	0.941	1.121	1.033	0.979	0.577

Table 1: Predictor variable means depending on whether the tissue samples were benign or malignant

The aim of this report is to is to build a classifier for the Class – benign or malignant – of a tissue sample based on (at least some of) the nine cytological characteristics. So it is very useful to look at the relationships between all the predictor variables. This can be done by creating pairwise plots (appendix 1.1) between all the explanatory variables:



Plot 1: Plot of pairwise plots of the predictor variables, where orange depicts benign tissues, and blue malignant

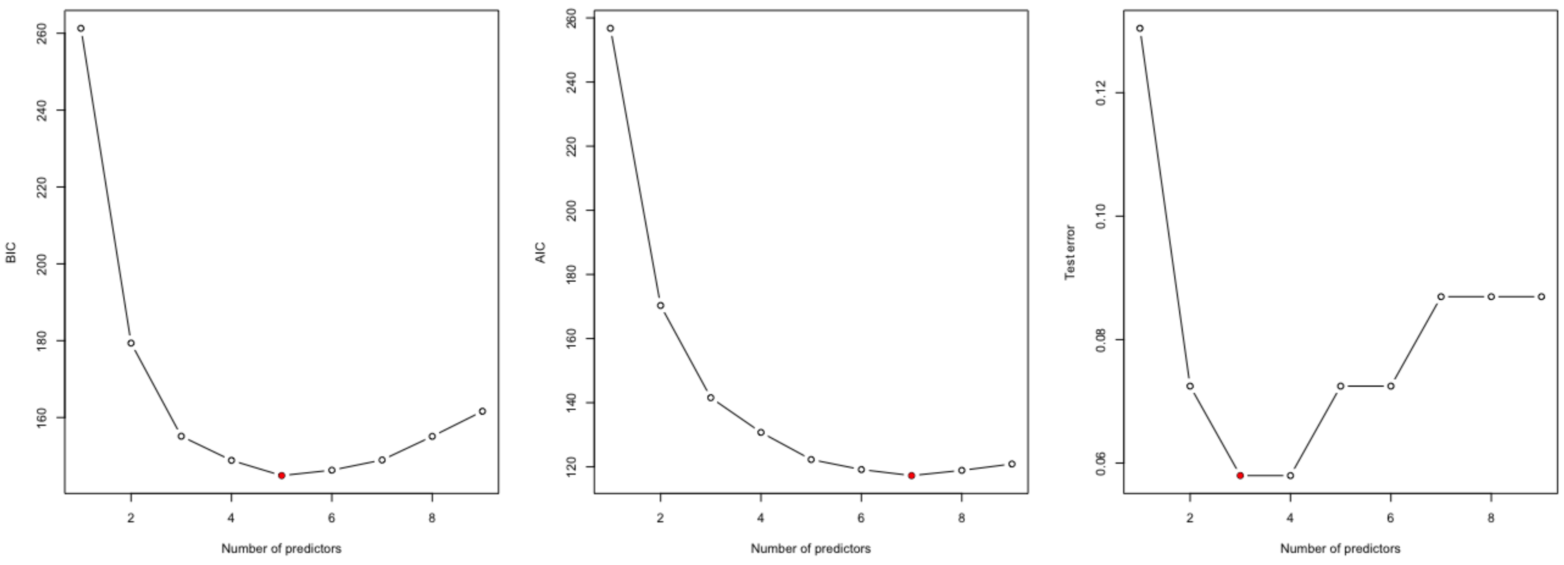
Looking at the plot and the correlations between each pair of variables, it is obvious that some predictor variables are highly correlated. For example the correlation between cell shape and cell size, is 0.907. This could be an indication that it is unlikely to need both variables in the model. Similarly we observe high correlations of around 0.7 between many variables, so again this could be an indication that not all 9 predictor variables are needed in the model. Finally, by looking at the correlations between the predictor variables and the response, it can be seen that the lowest correlation (0.423) is that between mitoses and y (Class), while the correlations of the other predictor variables with y are consistently high.
In order to build a classifier for y (Class variable) and decide on the number of predictor variables needed methods like subset selection, rigde and LASSO regression as well as discriminant analysis are going to be used. In addition, the performance off all the possible models found is going to be compared using 10 fold validation, with the same folds (appendix 2.2). Then the test error will be used for comparison.

Subset Selection

A very useful method to reduce the dimensions of a data set is subset selection. Using subset selection, helps in identifying a best model, by fitting all possible subsets of predictor variables. Here since the number of explanatory variables is not large, "exhaustive" subset selection, is chosen over "stepwise" since the method considers all possible models, and not just some of them. In order to identify the best model, three different decision criteria were used. These were the BIC, the AIC and the mean squared error. First apply best subset selection using the two criteria:

```r
## Apply the best subset selection method using BIC
bss_BIC_fit = bestglm(Breast_Cancer_data, family = binomial, method="exhaustive", nvmax=p)

## Apply the best subset selection method using AIC
bss_AIC_fit = bestglm(Breast_Cancer_data, family = binomial, method="exhaustive", nvmax=p, IC = "AIC")
```

In order to find the test error, cross validation is performed to the models identified by subset selection and the model with the smallest error is selected. This process can be done by creating a function (appendix 1.3) that performs 10 fold cross validation for each possible model, then finds the cross validation error when using each fold as a test set, and then final cross validation error, by the weighted mean of the individual cv errors. This is done for every model identified by best subset selection. In order to get a better idea of which model each criterion selects, it is useful to plot the three graphs below. The model selected by each criterion is denoted by the red dot:



Plot 2: Plots of the number of predictor variables against BIC, AIC, and Test error

From the plots, it is observed that the different criteria, select different models. BIC selects the model with 5 predictor variables, AIC the model with 7 predictor variables, while based on the lowest test error, the model with 3 variables is selected. However, looking at the test error, it is clear that model 4 has very similar test error as model 3. Similarly from the other two graphs, BIC and AIC for model 4, is not that different to the models selected by these criteria. So a good choice of model could be the one with 4 predictor variables. This model includes cell thickness, cell shape, bare nuclei and Bl. cromatin. The test error for this model is 0.05797101, and the coefficients of the predictor variables are:

```
              Estimate
(Intercept)  -1.142954
Cl.thickness  1.593822
Cell.shape    1.729167
Bare.nuclei   1.560146
Bl.cromatin   1.451359
```

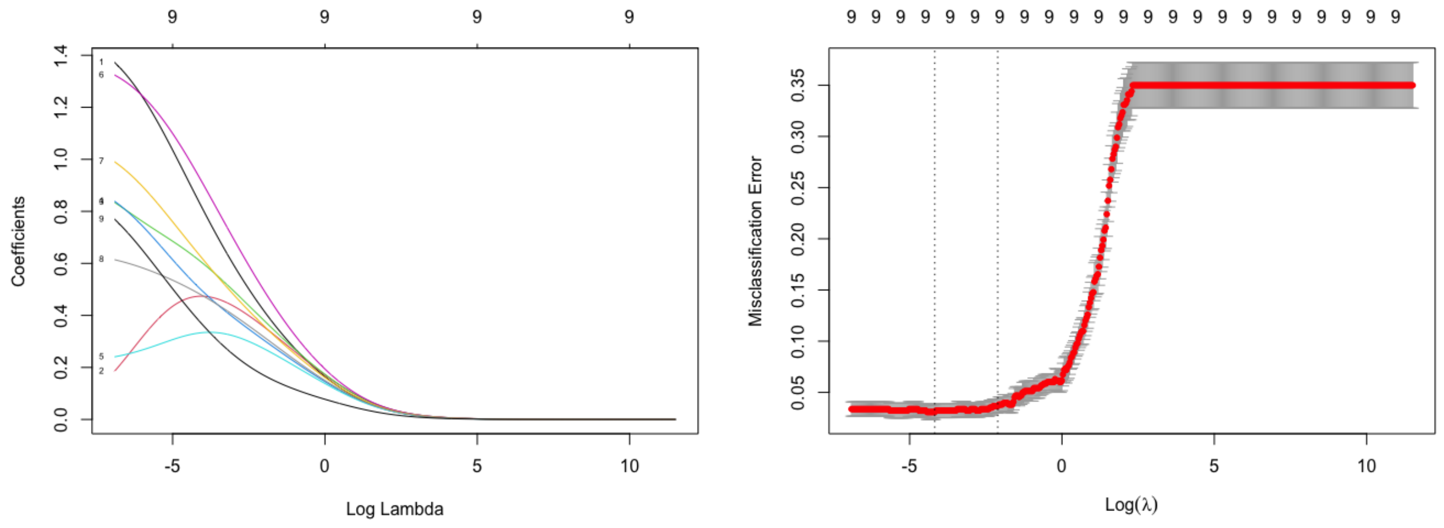
Cell thickness and cell shape have the higher coefficients.

Regularisation methods

Regularisation methods, are used in order to shrink the coefficients of the variables towards 0, or even make them equal to zero. For this analysis both ridge regression and the LASSO are considered.

1. Ridge Regression:

To perform ridge regression, a grid of values for the tuning parameter lamda needs to be selected and then the ridge regression model needs to be fitted for every value of lamda (appendix 1.4). In order to see how the coefficients of the predictor variables behave, a plot of lof lamda against the coefficients is useful. Moreover, in order to select the best value for lamda, cross validation is performed using the same folds as before (appendix 1.4). Then log lamda against misclassification error is plotted and the value of lamda that produces the lowest error is selected. The two plots discussed are plotted below:



Plot 3: Plots of log lamda against coefficients and misclassification error

Now from the plot on the left, it is clear that all the coefficients have shrunk to zero by the time that log lamda is 6. From the right plot, the minimum value of lamda is between the two doted lines. In order to identify this value the command below is used:

```
210  (lambda_ridge_min = ridge_cv_fit$lambda.min)
```

So the value of lamda that gives the lowest test error is: 0.01535935

Now fitting the model using the value of lamda identified above, gives the following coefficients:

```
> coef(ridge_fit, s=lambda_ridge_min)
10 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept)     -0.9743961
Cl.thickness     0.8458049
Cell.size        0.4733421
Cell.shape       0.6159912
Marg.adhesion    0.5111757
Epith.c.size     0.3316143
Bare.nuclei      0.9413250
Bl.cromatin      0.6398961
Normal.nucleoli  0.4864021
Mitoses          0.3848584
> 
```

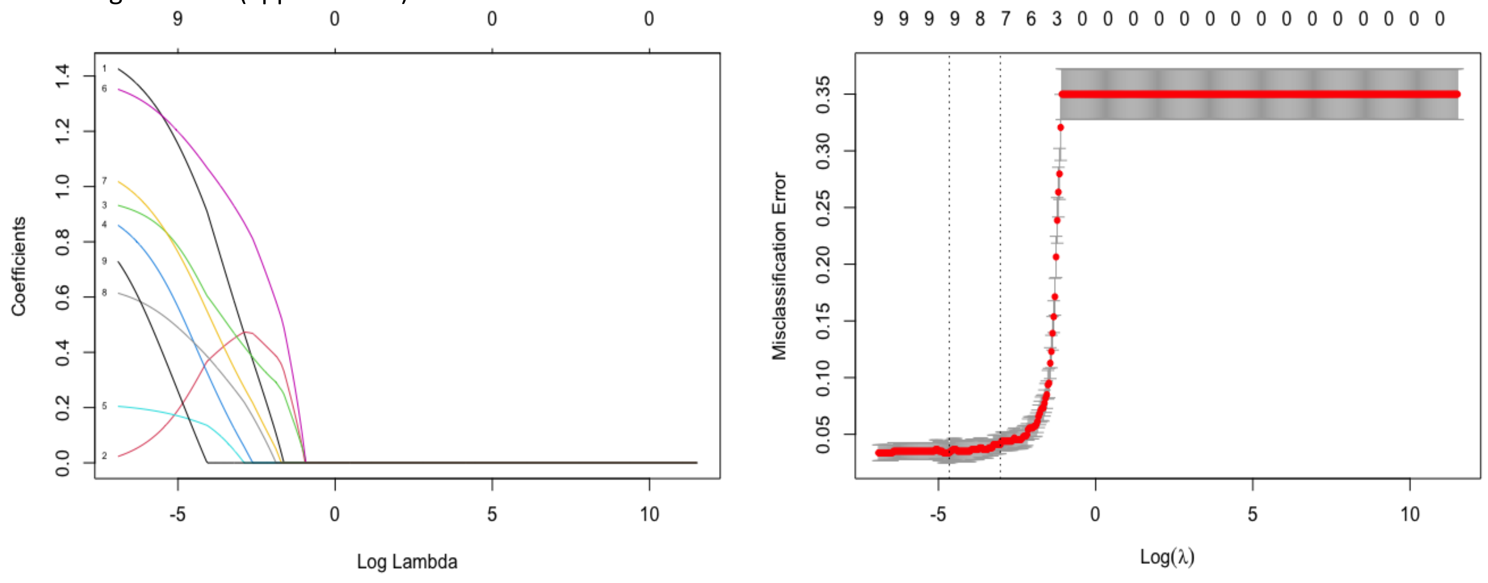
Similarly the coefficients when using the glm function are:

```
> glm_fit = glm(y ~ .,
+                data = Breast_Cancer_data, family = binomial)
> glm_fit$coefficients
    (Intercept)    Cl.thickness       Cell.size      Cell.shape   Marg.adhesion     Epith.c.size
    -1.09414457      1.50914699     -0.01924824      0.96443444      0.94713001       0.21482879
    Bare.nuclei     Bl.cromatin Normal.nucleoli         Mitoses
     1.39568683      1.09547471      0.65031161      0.92669587
```

Comparing the coefficients, it is clear that some of the parameter estimates have shrunk towards zero. However some others like cell size and epith.c size have increased.
Now the test error for this model is 0.03074671 (appendix 1.4). Compared with the test error for the model identified by best subset selection, it is clear that the test error has reduced.

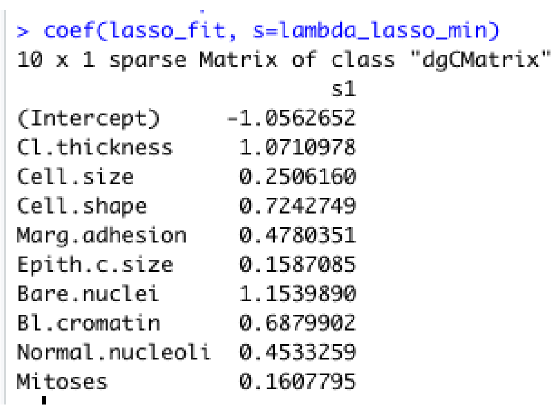
2. LASSO

Ridge regression shrinks the coefficients towards zero, but never really makes any coefficients equal to zero. That is why the LASSO can be very useful. Just like before the LASSO regression is fitted and the appropriate value for lamda is chosen, using cross validation. Then the corresponding plots are generated (appendix 1.5):



Plot 4: Plots of log lamda against coefficients and misclassification error for the LASSO

From the plot on the left, it is observed that the coefficient of some predictor variables go to zero earlier compared to others. So the variables that drop out of the model first are mitoses, followed by epith.c size and marg. adhesion. The last variables to drop out are cell size, cell shape and Bl. cromatin. From the right plot the minimum value of lamda can be found between the doted lines. Just like before this value is extracted (appendix 1.5) and is found to be 0.009505083. Fitting the model using this value of lamda gives the following coefficients:



```
> coef(lasso_fit, s=lambda_lasso_min)
10 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept)     -1.0562652
Cl.thickness     1.0710978
Cell.size        0.2506160
Cell.shape       0.7242749
Marg.adhesion    0.4780351
Epith.c.size     0.1587085
Bare.nuclei      1.1539890
Bl.cromatin      0.6879902
Normal.nucleoli  0.4533259
Mitoses          0.1607795
```

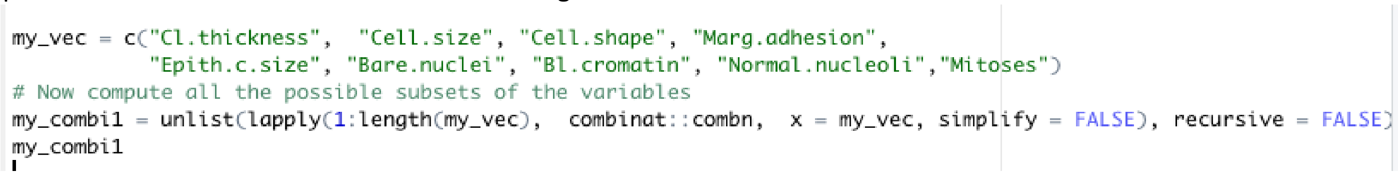
Now all coefficients have shrunk towards zero. However none of them is equal to zero. The test error for this model is 0.03367496 (appendix 1.5). Compared to the model identified by best subset selection it is again smaller. Now when compared with the one of ridge regression, it is slightly bigger, but does not differ much.

<u>Discriminant Analysis</u>

Discriminant analysis can be used for classification. Since the aim of this project is to find the best classifier, for the variable "Class", it is very useful to perform classification using both linear and quadratic analysis. These two methods differ in terms on some assumptions that are made for the conditional distributions for the predictor variables.

<u>1. Linear discriminant analysis (LDA)</u>

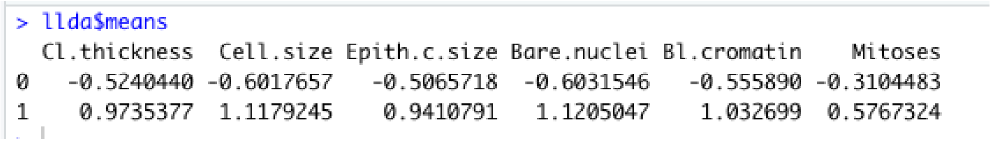
LDA is the first of the two methods to use for classification. In order to choose the optimal classification, all the possible subsets of the predictor variables will be consider, and the subset which minimises the cross validation estimate of the test error will be chosen. Finding all the possible subsets can be done with the following commands:



```
my_vec = c("Cl.thickness",  "Cell.size", "Cell.shape", "Marg.adhesion",
           "Epith.c.size", "Bare.nuclei", "Bl.cromatin", "Normal.nucleoli","Mitoses")
# Now compute all the possible subsets of the variables
my_combi1 = unlist(lapply(1:length(my_vec),  combinat::combn,  x = my_vec, simplify = FALSE), recursive = FALSE)
my_combi1
```

Now everything is ready in order to perform LDA. In order to perform LDA on all possible subsets and then calculating the test error, a function is required (appendix 1.6). This function loops over all possible subsets, performs LDA in each one of them, and then finds the test error. Then the subset with the smallest test error is selected. After running the function, the model that gives the lowest test error is found to be the one that includes Cell thickness, Cell size, Epith.c. size, Bare. nuclei, Bl.cromatin and Mitoses as it's explanatory variables. The test error for this model is found to be 0.03660322. Compared to the errors found by ridge regression and LASSO, there are not very significant differences. However, it is a bit higher, so LDA performs a bit worse than ridge regression and the LASSO.

Now that the model with the lowest test error has been found, LDA can be performed (appendix 1.6) in order to get a better insight of the different predictor variable means (appendix 1.6) for the different groups. So the means are found to be:



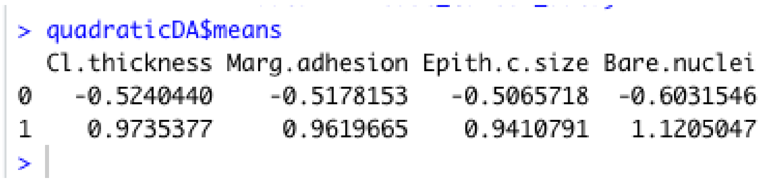
```
> llda$means
  Cl.thickness  Cell.size Epith.c.size Bare.nuclei Bl.cromatin    Mitoses
0   -0.5240440 -0.6017657   -0.5065718  -0.6031546   -0.555890 -0.3104483
1    0.9735377  1.1179245    0.9410791   1.1205047    1.032699  0.5767324
```

Figure 1: Predictor variable means (using LDA) for tissue classified as benign (top) and as malignant (bottom)

Observing the predictor variable means, it is clear that those for tissues classified as benign are all negative, and thus significantly lower than those for the tissues classified as malignant. So in general, it is expected that tissues which will be classified as malignant, will exhibit higher values to these 6 predictor variables.

2. Quadratic Discriminant analysis

QDA is the second method used for classification. The process of finding the model with the lowest test error, is very similar to the one used before. There are however some differences in the function that was used before (appendix 1.7). So the model that was found to produce the smallest test error was the one that included Cell thickness, Marg. adhesion, Epith.c. size and Bare. nuclei. Now the test error for this model was found to be 0.03513909. Compared to LDA, the error is slightly lower, so for this data set it seems that QDA performs a bit better than LDA. Now again, it would be very useful to look at the predictor variable means (appendix 1.7) for the different classes. These are shown below:



```
> quadraticDA$means
  Cl.thickness Marg.adhesion Epith.c.size Bare.nuclei
0   -0.5240440    -0.5178153   -0.5065718  -0.6031546
1    0.9735377     0.9619665    0.9410791   1.1205047
>
```

Figure 2: Predictor variable means (using QDA) for tissue classified as benign (top) and as malignant (bottom)

Again when observing the means, it is clear that the tissues classified as benign tend to have lower means than the ones of the tissues classified as malignant. So just like before, it is expected that tissues which will be classified as malignant, will exhibit higher values to these 4 predictor variables.

Comparison

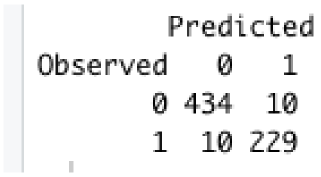
Different ways, have been used in order to build the best classifier. As a measure for comparison between the models identified by each classifier, the cross validation test error will be used. The results are summarised in the table below:

Predictor	Best Subset Selection(4)	Ridge Regression	LASSO	LDA	QDA
Intercept	- 1.143	-0.974	-1.056		
Cell thickness	1.594	0.846	1.071	included	included
Cell size	.	0.473	0.251	Included	
Cell shape	1.729	0.616	0.724		
Marg.adhesion	.	0.511	0.478		included
Epith.c.size	.	0.332	0.159	included	included
Bare. Nuclei	1.560	0.941	1.154	included	included
Bl. cromatin	1.451	0.640	0.688	included	
Normal nucleoni	.	0.486	0.453		
Mitoses	.	0.385	0.161	included	
Test error	**0.0580**	**0.0307**	**0.0337**	**0.0366**	**0.0351**

Table 2 : Table comparing the different models, using the test error

The model which minimises the test error is the ridge regression model, which contains all of the predictor variables. This is very similar to the LASSO model which also contains all variables. Moreover LDA and QDA also give similar test errors to the selected model.
Now looking at the ridge regression model, the most dominant predictor variables are cell thickness and bare nuclei. Their coefficients are positive, suggesting that tissues with higher cell thickness and bare nuclei measurements are more possible to be classified as malignant. In general all the coefficients for the chosen classifier are positive, meaning that the higher the measurements on these predictor variables, the more possible for the tissue to be classified as malignant.

Finally, for the chosen model it is very important to identify the type of error that it tends to make. In order to identify the misclassification error the table below (appendix 1.8) is created:



```
        Predicted
Observed   0   1
       0 434  10
       1  10 229
```

Table 3: Table showing the misclassification error of the chosen model.

The chosen model seems to classify correctly as benign 434 of 444 tissues and correctly as malignant 229 of the 239 tissues. Proportionally it is observed that 97.7% of the benign tissues are classified correctly as benign, while 95.8% of malignant tissues are classified correctly as malignant. So in general the model tends to classify malignant tissues as benign more often that the other way around.

Conclusion

After assessing different methods for building a classifier for the “Class” variable, the ridge regression model was selected among the models identified by best subset selection, the LASSO, LDA and QDA. The selection was based on the test error, and the model with the lowest error was selected. The ridge regression model, includes all of the predictor variables and it tends to misclassify malignant tissues more often, when compared to classifying benign tissues.

1.

Appendix

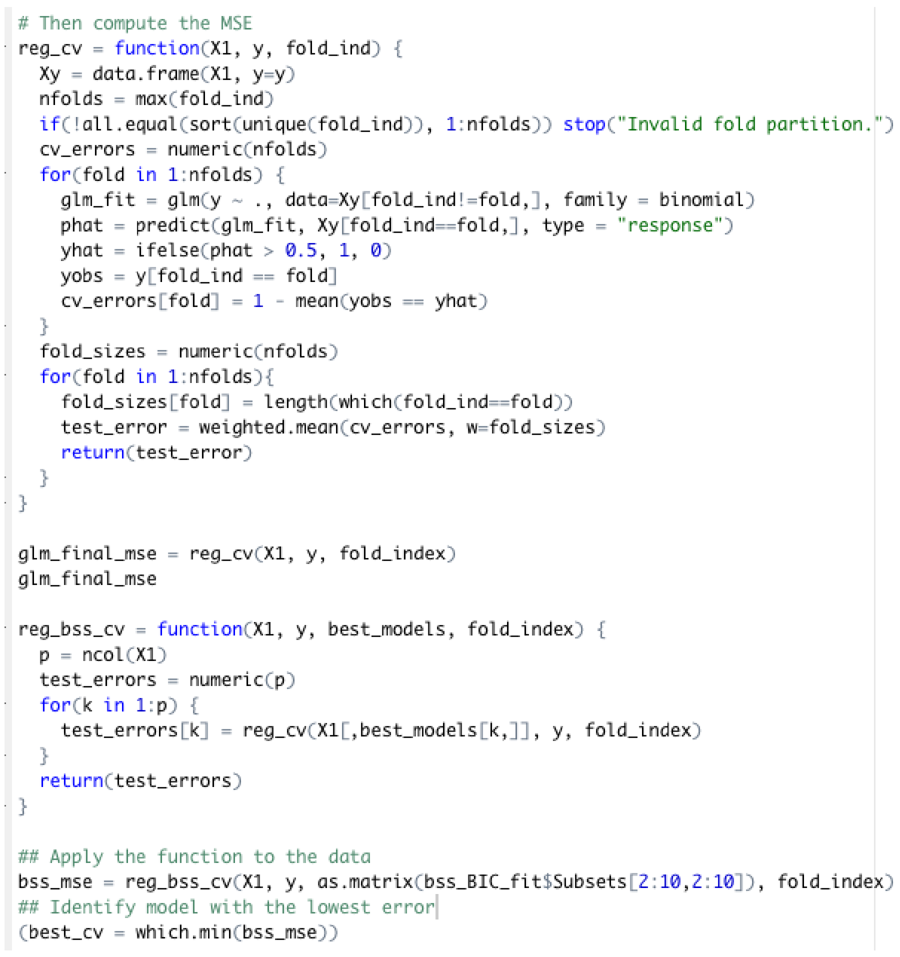
1.1

```
#Check size
dim(BreastCancer)
head(BreastCancer)
sapply(BreastCancer,class)

#choosing which columns we want to change into variables
i = c(1:10)
# converting the factors to variables
#We need to do that because most of the statistical concepts  can only be applied
BreastCancer[ , i] = apply(BreastCancer[ , i], 2, function(x) as.numeric(as.character(x)))
#Checking the class of each column
sapply(BreastCancer,class)

#Now deleting the rows with NA values and creating a new data frame
NewBreastCancer = na.omit(BreastCancer[, 2:11])
dim(NewBreastCancer)
sapply(NewBreastCancer,class)

# Explaratory analysis
install.packages("GGally")
library(GGally)
ggpairs(Breast_Cancer_data, lower = list(combo = wrap("facethist", bindwidth = 0.5)), aes(colour = as.factor(y)))
```

1.2

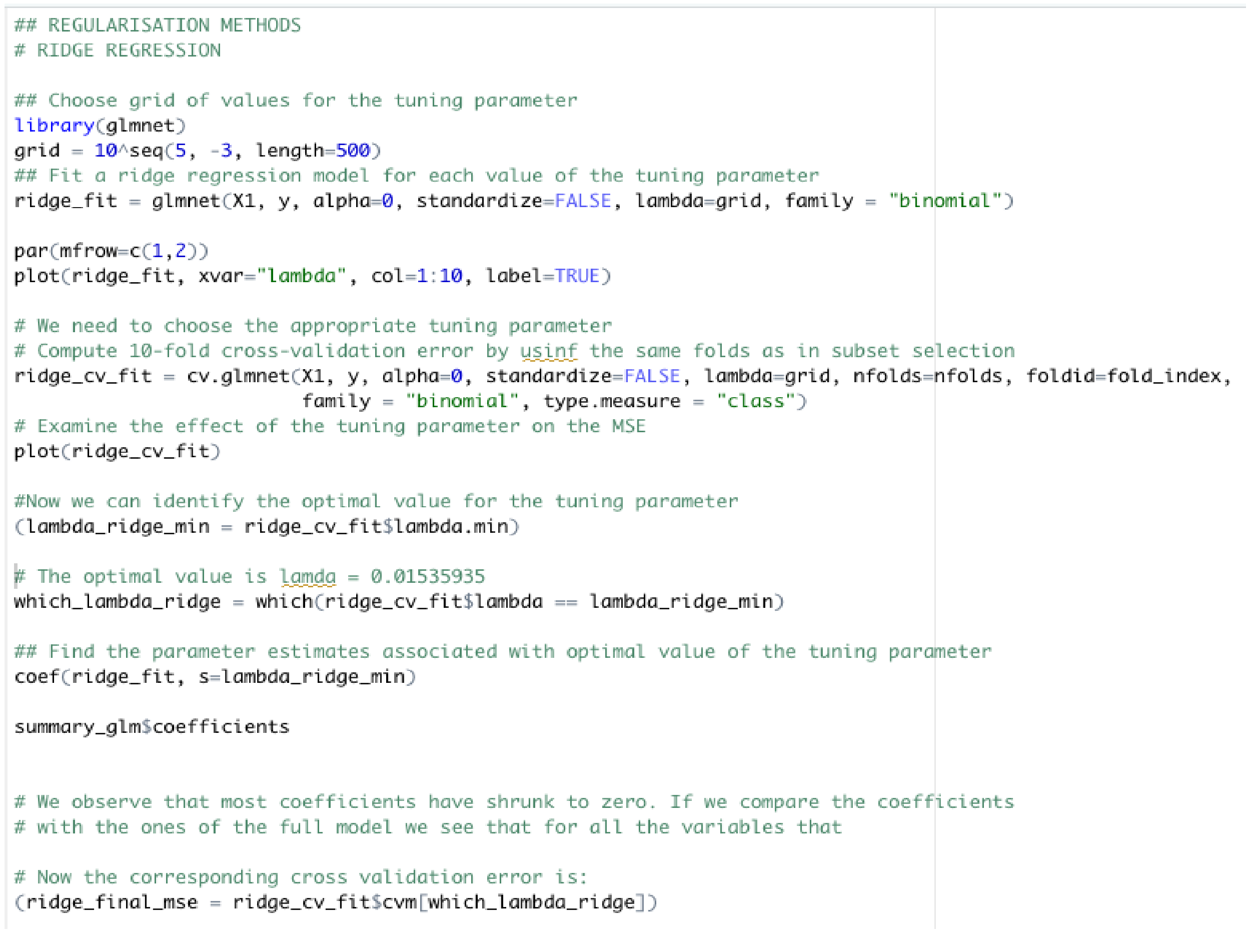
```
# Extract response variable
# We also need to convert the class to a numeric factor where 1 is for benign and 2 for
#malignant
y = as.numeric(NewBreastCancer[,10]) -1
# Extract predictor variables
X1orig = NewBreastCancer[,-10]
# Standardise predictor variables
X1 = scale(X1orig)
## Combine response and standardised predictors variables in a new data frame
Breast_Cancer_data = data.frame(X1,y)
head(Breast_Cancer_data)
p = ncol(X1)
n = nrow(X1)
table(Breast_Cancer_data$y)

round(apply(Breast_Cancer_data[Breast_Cancer_data[, "y"] == 0,], 2, mean),3)
round(apply(Breast_Cancer_data[Breast_Cancer_data[, "y"] == 1,], 2, mean),3)
# We will compare the predictive performance of the different techniques using
#10-fold cross-validation. To make the comparison fair, we will use the same set of 10 folds
#in each case. We can sample and store the 10-folds as follows:

set.seed(1)
nfolds = 10
## Set the seed to make the analysis reproducible

## Sample fold-assignment index
fold_index = sample(nfolds, n, replace=TRUE)
```

1.3



```
# Then compute the MSE
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) {
    glm_fit = glm(y ~ ., data=Xy[fold_ind!=fold,], family = binomial)
    phat = predict(glm_fit, Xy[fold_ind==fold,], type = "response")
    yhat = ifelse(phat > 0.5, 1, 0)
    yobs = y[fold_ind == fold]
    cv_errors[fold] = 1 - mean(yobs == yhat)
  }
  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)
  }
}

glm_final_mse = reg_cv(X1, y, fold_index)
glm_final_mse

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)
}

## Apply the function to the data
bss_mse = reg_bss_cv(X1, y, as.matrix(bss_BIC_fit$Subsets[2:10,2:10]), fold_index)
## Identify model with the lowest error
(best_cv = which.min(bss_mse))
```

1.4



```
## REGULARISATION METHODS
# RIDGE REGRESSION

## Choose grid of values for the tuning parameter
library(glmnet)
grid = 10^seq(5, -3, length=500)
## Fit a ridge regression model for each value of the tuning parameter
ridge_fit = glmnet(X1, y, alpha=0, standardize=FALSE, lambda=grid, family = "binomial")

par(mfrow=c(1,2))
plot(ridge_fit, xvar="lambda", col=1:10, label=TRUE)

# We need to choose the appropriate tuning parameter
# Compute 10-fold cross-validation error by usinf the same folds as in subset selection
ridge_cv_fit = cv.glmnet(X1, y, alpha=0, standardize=FALSE, lambda=grid, nfolds=nfolds, foldid=fold_index,
                         family = "binomial", type.measure = "class")
# Examine the effect of the tuning parameter on the MSE
plot(ridge_cv_fit)

#Now we can identify the optimal value for the tuning parameter
(lambda_ridge_min = ridge_cv_fit$lambda.min)

# The optimal value is lamda = 0.01535935
which_lambda_ridge = which(ridge_cv_fit$lambda == lambda_ridge_min)

## Find the parameter estimates associated with optimal value of the tuning parameter
coef(ridge_fit, s=lambda_ridge_min)

summary_glm$coefficients


# We observe that most coefficients have shrunk to zero. If we compare the coefficients
# with the ones of the full model we see that for all the variables that

# Now the corresponding cross validation error is:
(ridge_final_mse = ridge_cv_fit$cvm[which_lambda_ridge])
```

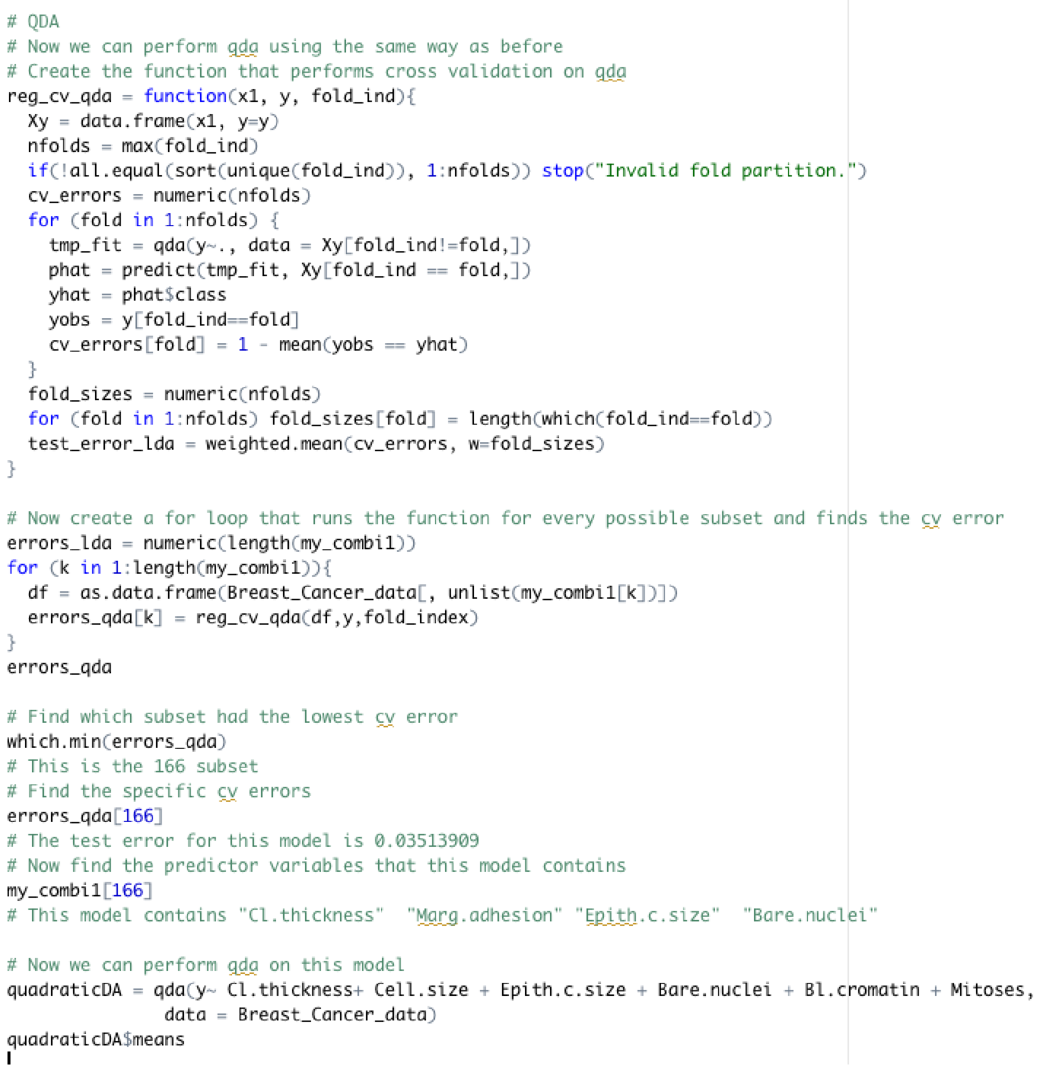
1.5

```
## THE LASSO
## Choose grid of values for the tuning parameter
grid1 = 10^seq(5, -3, length=500)
## Fit a LASSO regression for each value of the tuning parameter
lasso_fit = glmnet(X1, y, alpha=1, standardize=FALSE, lambda=grid1, family = "binomial")

## Examine the effect of the tuning parameter on the parameter estimates
plot(lasso_fit, xvar="lambda", col=1:10, label=TRUE)

# Examining the plot we see that variable 9 is the first to
# drop out, followed by variables 5 and 9. The last variables to drop out are 2, 3 and 6

# Compute 10-fold cross-validation error using the same folds as in ss and ridge regression
lasso_cv_fit = cv.glmnet(X1, y, alpha=1, standardize=FALSE, lambda=grid, nfolds=nfolds, foldid=fold_index,
                         family = "binomial", type.measure = "class")
plot(lasso_cv_fit)

## Identify the optimal value for the tuning parameter
(lambda_lasso_min = lasso_cv_fit$lambda.min)
which_lambda_lasso = which(lasso_cv_fit$lambda == lambda_lasso_min)
## Find the parameter estimates associated with optimal value of the tuning parameter
coef(lasso_fit, s=lambda_lasso_min)

# We see that the coefficients for id has shrunk to zero. However for all
# the other coefficients we do not observe any significant reduction

# Now the corresponding cross validation error is:
(lasso_final_mse = lasso_cv_fit$cvm[which_lambda_lasso])
```

1.6

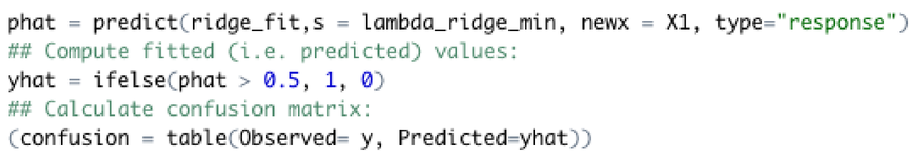
```
install.packages("MASS")
library(MASS)

# Create the function that performs cross validation on lda
reg_cv_lda = 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 = lda(y~., data = Xy[fold_ind!=fold,])
    phat = predict(tmp_fit, Xy[fold_ind == fold,])
    yhat = phat$class
    yobs = y[fold_ind==fold]
    cv_errors[fold] = 1 - mean(yobs == yhat)
  }
  fold_sizes = numeric(nfolds)
  for (fold in 1:nfolds) fold_sizes[fold] = length(which(fold_ind==fold))
  test_error_lda = weighted.mean(cv_errors, w=fold_sizes)
}

# Now create a for loop that runs the function for every possible subset and finds the cv error
errors_lda = numeric(length(my_combi1))
for (k in 1:length(my_combi1)){
  df = as.data.frame(Breast_Cancer_data[, unlist(my_combi1[k])])
  errors_lda[k] = reg_cv_lda(df,y,fold_index)
}
errors_lda

# Find which subset had the lowest cv error
which.min(errors_lda)
# This is the 413 subset
# Find the specific cv errors
errors_lda[413]
# The test error for this model is 0.03660322
# Now find the predictor variables that this model contains
my_combi1[413]
# So this model contains "Cl.thickness" "Cell.size"    "Epith.c.size" "Bare.nuclei"  "Bl.cromatin"  "Mitoses"

# Now we can perform LDA and find the group means
llda = lda(y~ Cl.thickness+ Cell.size + Epith.c.size + Bare.nuclei + Bl.cromatin + Mitoses,
           data = Breast_Cancer_data)
llda$means
```

1.7



```
# QDA
# Now we can perform qda using the same way as before
# Create the function that performs cross validation on qda
reg_cv_qda = 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 = qda(y~., data = Xy[fold_ind!=fold,])
    phat = predict(tmp_fit, Xy[fold_ind == fold,])
    yhat = phat$class
    yobs = y[fold_ind==fold]
    cv_errors[fold] = 1 - mean(yobs == yhat)
  }
  fold_sizes = numeric(nfolds)
  for (fold in 1:nfolds) fold_sizes[fold] = length(which(fold_ind==fold))
  test_error_lda = weighted.mean(cv_errors, w=fold_sizes)
}

# Now create a for loop that runs the function for every possible subset and finds the cv error
errors_lda = numeric(length(my_combi1))
for (k in 1:length(my_combi1)){
  df = as.data.frame(Breast_Cancer_data[, unlist(my_combi1[k])])
  errors_qda[k] = reg_cv_qda(df,y,fold_index)
}
errors_qda

# Find which subset had the lowest cv error
which.min(errors_qda)
# This is the 166 subset
# Find the specific cv errors
errors_qda[166]
# The test error for this model is 0.03513909
# Now find the predictor variables that this model contains
my_combi1[166]
# This model contains "Cl.thickness"  "Marg.adhesion" "Epith.c.size"  "Bare.nuclei"

# Now we can perform qda on this model
quadraticDA = qda(y~ Cl.thickness+ Cell.size + Epith.c.size + Bare.nuclei + Bl.cromatin + Mitoses,
               data = Breast_Cancer_data)
quadraticDA$means
```

1.8



```
phat = predict(ridge_fit,s = lambda_ridge_min, newx = X1, type="response")
## Compute fitted (i.e. predicted) values:
yhat = ifelse(phat > 0.5, 1, 0)
## Calculate confusion matrix:
(confusion = table(Observed= y, Predicted=yhat))
```

Theoretical Part:

We need to show that the logistic regression with the classification rule:

$$Y = \begin{cases} 1, & P(Y=1 \mid \underline{X} = \underline{x}) > \alpha \\ 0, & \text{otherwise} \end{cases}$$

is equivalent to a discriminant rule with a linear boundary between two allocation groups:

Now we would like to construct a rule where we allocate to group 1 if $P(Y=1 \mid \underline{x} = \underline{x}) > \alpha$.

Now in logistic regression we know that:

$$P(Y=1 \mid \underline{x} = \underline{x}) = \frac{\exp(\beta_0 + \beta_1 x_1 + \dots \beta_p x_p)}{1 + \exp(\beta_0 + \beta_1 x_1 + \dots \beta_p x_p)} = \frac{e^n}{1 - e^n}$$

where $n = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p$

So we want:

$$P(Y=1 \mid \underline{X} = \underline{x}) > \alpha$$

$$\frac{e^n}{1 - e^n} > \alpha$$

So for the logistic regression we have:

$$Y = \begin{cases} 1, & \text{if } \frac{e^n}{1 - e^n} > \alpha \\ 0, & \text{otherwise} \end{cases}$$

as required ■