MAS8383 Statistical Learning Methodology Project

Luke Kaye

1.1 Applied Part

We have a dataset, *BreastCancer*, containing 699 observations on 11 variables relating to breast tissue samples collected from women. The purpose of the experiment that gave rise to the data was to determine the extent to which a tissue sample could be classified as *benign* or *malignant* using only the nine cytological characteristics described in the dataset.

Our first data column, *Id*, is a character variable, referring to the patient that the observation corresponds to. The following nine data columns are each ordinal variables on a 1-10 scale, each describing a cytological characteristic of the tissue sample. The final column, *Class*, is a factor with two levels, *benign* and *malignant*. We are interested in all the variables within the dataset, excluding *Id*.

Data Cleaning

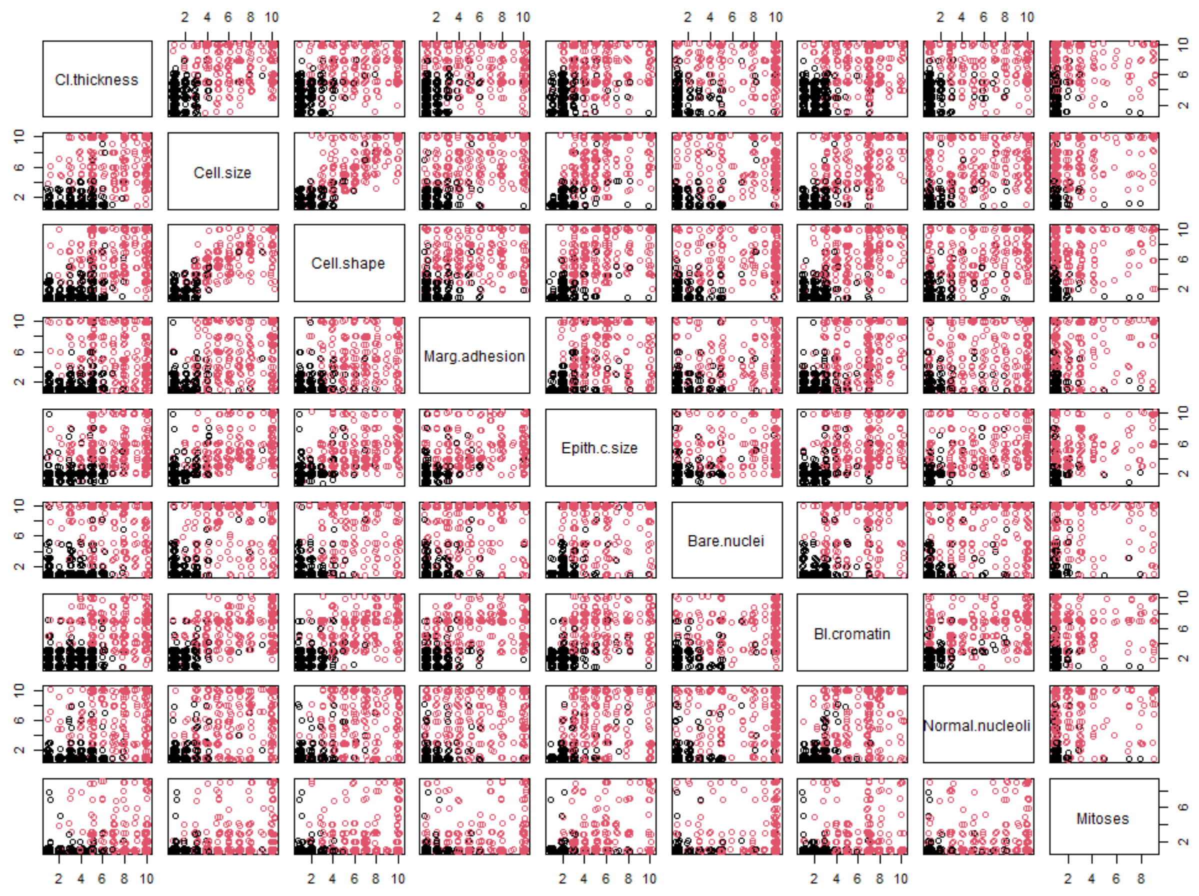
Our nine cytological characteristics are currently coded as factors. Later, we will be considering classification models for our dataset, one such model will be produced through a discriminant analysis method. A model of this type requires all predictors to be continuous due to the assumption of multivariate normality of the predictors, therefore we will convert our cytological characteristics to quantitative variables. Making this conversion is reasonable since the cytological characteristics are on a scale relative to healthiness, a concept which is naturally continuous, and therefore it is logical that values interpolated between the discrete levels given in the dataset would be possible. In other words, the 1-10 scale is arbitrary in its ten levels; it would be equally plausible to rank the characteristics on a different scale, say with twenty levels, which would then allow values of, for example, 1.5 or 5.5.

Using the following R code, the conversion was made and checked, then any observations containing a missing value in one or more of its variables were removed, leaving us with 683 observations to analyse.

```
# convert cytological characteristics to quantitative variables
BreastCancer[, 2:10] = lapply(BreastCancer[, 2:10], as.numeric)
# check conversion has been applied successfully
lapply(BreastCancer, class)
# remove rows containing missing values
BreastCancer = na.omit(BreastCancer)
```

Exploratory Data Analysis

We will begin by viewing a scatterplot matrix of the predictors. As the data is discrete, we will use the *jitter* function within R to add a small amount of random noise to each observation on the plot. This gives a measure of density to each discrete value on each plot within the matrix.



The perceived colour density to the plots allows us to see the concentration of values at each discrete point, which was not possible before. It additionally allows us to see how a point is distributed between its *benign* and *malignant* factors.

Benign tissue samples correspond to black points, whereas *malignant* tissue samples correspond to red points. We see that *benign* tissue samples are largely clustered around smaller values of the predictors, with occasional outliers. The spread of *malignant* tissue samples is much greater, dominating each plot outside of the low-value areas.

We can immediately see that each predictor has a large degree of variance relative to other predictors. There is a strong positive correlation between *Cell.size* and *Cell.shape*, thus including both in our eventual classifier may be unnecessary. There is evidence of other correlations between our predictors, such as between *Epith.c.size* and both *Cell.size* and *Cell.shape* (again, due to the strong correlation between *Cell.size* and *Cell.shape*, we would expect both of these variables to share correlations with other predictors). *Epith.c.size* also seems to share some correlative behaviour with *Marg.adhesion* and *Bl.cromatin* as well. *Cl.thickness* shows a positive correlation with *Cell.size* and *Cell.shape*, and possibly with *Epith.c.size*. Beyond this, it is hard to make out any correlative behaviour between predictors.

We also note the count of *benign* and *malignant* samples:

```
   benign malignant
      444       239
```

We see that there are roughly twice as many *benign* samples as there are *malignant*. The counts of each should allow for sufficient accuracy in building a classifier, in terms of prospective training set sample size.

We will now look at the sample mean vector and sample covariance matrix for the data to see if standardising our data will be necessary for our classification models. This is an important consideration for regularised regression to ensure that all predictors are penalised evenly.

We compute the sample mean vector of the predictors:

```
  Cl.thickness      Cell.size     Cell.shape  Marg.adhesion   Epith.c.size
      4.442167       3.150805       3.215227       2.830161       3.234261
   Bare.nuclei    Bl.cromatin Normal.nucleoli        Mitoses
      3.544656       3.445095        2.869693       1.582723
```

While it appears that the means all seem to be very similar and close to zero, we must take into consideration that each predictor can only take a value between 1 and 10 inclusive. In this regard, we have a spectrum of mean values that covers just under a third of the range of predictor values. This is certainly not negligible and thus centering should be considered.

We compute the sample variances and sample correlation matrix of the predictors.

Sample variances:

```
   Cl.thickness       Cell.size      Cell.shape   Marg.adhesion    Epith.c.size     Bare.nuclei     Bl.cromatin Normal.nucleoli
       7.956694        9.395113        8.931615        8.205717        4.942109       13.277695        6.001013        9.318772
        Mitoses
       2.677531
```

Our sample variances range from approximately 2.7 to 13.3. Again, this sounds like a small range however each predictor takes values in the inclusive range 1-10 and therefore this spectrum of variances is significant, suggesting a full standardisation would be appropriate as opposed to simply centering the data.

Sample correlation matrix:

```
                Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei Bl.cromatin Normal.nucleoli   Mitoses
Cl.thickness       1.0000000 0.6424815  0.6534700     0.4878287    0.5235960   0.5930914   0.5537424       0.5340659 0.3545301
Cell.size          0.6424815 1.0000000  0.9072282     0.7069770    0.7535440   0.6917088   0.7555592       0.7193460 0.4654091
Cell.shape         0.6534700 0.9072282  1.0000000     0.6859481    0.7224624   0.7138775   0.7353435       0.7179634 0.4468571
Marg.adhesion      0.4878287 0.7069770  0.6859481     1.0000000    0.5945478   0.6706483   0.6685671       0.6031211 0.4249917
Epith.c.size       0.5235960 0.7535440  0.7224624     0.5945478    1.0000000   0.5857161   0.6181279       0.6289264 0.4811836
Bare.nuclei        0.5930914 0.6917088  0.7138775     0.6706483    0.5857161   1.0000000   0.6806149       0.5842802 0.3490108
Bl.cromatin        0.5537424 0.7555592  0.7353435     0.6685671    0.6181279   0.6806149   1.0000000       0.6656015 0.3536683
Normal.nucleoli    0.5340659 0.7193460  0.7179634     0.6031211    0.6289264   0.5842802   0.6656015       1.0000000 0.4370424
Mitoses            0.3545301 0.4654091  0.4468571     0.4249917    0.4811836   0.3490108   0.3536683       0.4370424 1.0000000
```

According to the sample correlation matrix, all the predictors have positive correlations, most of which are substantial. This could indicate multicollinearity so we should proceed our analysis with caution. The correlation between *Cell.size* and *Cell.shape* is approximately 0.9 which reinforces what we saw in the scatterplot matrix. Beyond that, we have positive correlations within the approximate range 0.35 – 0.75. The correlations indicate a strong chance we won't need all the predictors in our final model.

Classification Modelling

Here we build three types of classification model for the data, each attempting to predict whether a sample is *benign* or *malignant*, based off the cytological characteristics described in the dataset:

1. Non-regularised logistic regression
    - We will use *Best Subset Selection* (BSS) to select a model of this type. We are considering 683 observations on 9 predictors which is feasible for BSS, despite being a computationally demanding method. Use of BSS over *Stepwise Selection* guarantees we choose the best possible model of this type.
2. Regularised logistic regression
    - We will use a *LASSO* penalty for this model to take advantage of its property for variable selection. This will allow better comparison with our non-regularised logistic regression model since BSS also allows for variable selection. Additionally, from what we have seen in the exploratory analysis, we should certainly be considering removing variables from our model.
3. Discriminant analysis
    - For a model of this type, it is not immediately obvious if a *Linear Discriminant Analysis* (LDA) or a *Quadratic Discriminant Analysis* (QDA) is superior for our dataset. When we consider a model of this type, we will analyse the first two principle components of the predictors to decide which to use.

We begin by standardising the predictors in the *BreastCancer* data. As each predictor is set on a dimensionless 1 – 10 scale, a scaling transformation will retain the property of low values indicating good health and high values indicating poor health. This standardised data will be used in all upcoming models.

Before we consider our classification models, we discuss the method we will use to assess model predictive error. For all our models, we will use 10-fold cross validation to select the model of each type with the lowest *misclassification probability*, thus selecting models with the lowest predictive error. Using the same type of test error for each model assists with the fairness in choosing a model. To allow reproducibility of this analysis, we will first use the *set.seed* function in R, setting its value to *1*. We then use the following code to divide the data into 10 folds. We will be using this fold classification for all model predictive error assessments, ensuring that differences in test error across models are not subject to individual choice in folds.

```
# cross validation folds, used for all models
fold_index = sample(rep(1:10, length = 683)) # n = 683
```

Non-Regularised Logistic Regression with Best Subset Selection

We utilise the *bestglm* function within the *bestglm* R library to perform BSS for the logistic regression models for our data. We firstly code our *Class* variable as a binary value, with *0 = benign* and *1 = malignant*, as binary coding is required by *bestglm*. We will retain this binary coding henceforth.

While *bestglm* has functionality for cross validation, we cannot define our own fold indices for use in the parameters of the function. Therefore, we will run *bestglm* on the full dataset, purely to find the best model corresponding to each number of predictors.

We run *bestglm,* seeing the following summary for the BSS algorithm:

```
# bss algorithm on 9 predictors
bss_fit = bestglm(BreastCancer[, 2:11], family = binomial)
```

```
bss_fit$Subsets
 Intercept Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses logLikelihood      BIC
      TRUE        FALSE     FALSE      FALSE         FALSE        FALSE       FALSE       FALSE           FALSE   FALSE    -442.17509 884.3502
      TRUE        FALSE      TRUE      FALSE         FALSE        FALSE       FALSE       FALSE           FALSE   FALSE    -127.37980 261.2861
      TRUE        FALSE      TRUE      FALSE         FALSE        FALSE        TRUE       FALSE           FALSE   FALSE     -83.15598 179.3649
      TRUE         TRUE      TRUE      FALSE         FALSE        FALSE        TRUE       FALSE           FALSE   FALSE     -67.77778 155.1351
      TRUE         TRUE     FALSE       TRUE         FALSE        FALSE        TRUE        TRUE           FALSE   FALSE     -61.37155 148.8491
      TRUE         TRUE     FALSE      FALSE          TRUE        FALSE        TRUE        TRUE            TRUE   FALSE     -56.13177 144.8960
      TRUE         TRUE     FALSE       TRUE          TRUE        FALSE        TRUE        TRUE            TRUE   FALSE     -53.57186 146.3027
      TRUE         TRUE     FALSE       TRUE          TRUE        FALSE        TRUE        TRUE            TRUE    TRUE     -51.63998 148.9654
      TRUE         TRUE     FALSE       TRUE          TRUE         TRUE        TRUE        TRUE            TRUE    TRUE     -51.45031 155.1126
      TRUE         TRUE      TRUE       TRUE          TRUE         TRUE        TRUE        TRUE            TRUE    TRUE     -51.44991 161.6383
```

We see that *bestglm* has selected which predictors to use for each model, each model providing the best fit for its number of predictors. We then proceed with our own cross validation method.

To compute the misclassification probability of each model, we define a set of functions:

```
# primary function, uses k-fold validation to find misclassification probability for a model
reg_cv = function(X1, y, fold_ind) {
  Xy = data.frame(X1, y=y) # get data frame formed of predictors and response
  nfolds = max(fold_ind) # get total number of folds
  if(!all.equal(sort(unique(fold_ind)), 1:nfolds)) stop("Invalid fold partition.")
  cv_errors = numeric(nfolds) # empty vector of cv errors
  for(fold in 1:nfolds) {
    tmp_fit = glm(y ~ ., data=Xy[fold_ind!=fold,], family = 'binomial') # fit predictive model
    yhat = predict(tmp_fit, Xy[fold_ind==fold,], type = 'response') # get predicted probs
    yhat_class = round(yhat) # round predicted probs to 1 or 0, i.e. get their classification
    yobs = y[fold_ind==fold] # find the actual classifications in the data set
    cv_errors[fold] = 1 - mean(yobs == yhat_class) # get misclassification probability overall
  }
  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) # weighted by fold sizes, getting overall error
  return(test_error)
}

# pass each BSS model into the above primary function, returning error for each
reg_bss_cv = function(X1, y, best_models, fold_index) {
  p = ncol(X1)
  test_errors = numeric(p)
  for(k in 1:p) { # pass each candidate model into the above function
    test_errors[k] = reg_cv(X1[,best_models[k,]], y, fold_index)
  }
  return(test_errors)
}
```

We then utilise these functions, using our previous fold indices, with

```
bss_err = reg_bss_cv(BreastCancer[, 2:10], BreastCancer[, 11],
                     as.matrix(bss_fit$Subsets[-1,2:10]), fold_index)
```

In short, the set of BSS models, returned by *bestglm* as Boolean values, are passed into *reg_bss_cv* which are then individually passed into *reg_cv*. Each time, the *BreastCancer* data are split into the same K-folds. The K-fold validation method is then used to compute a weighted average misclassification probability for each model, which are each stored as a numeric vector.

We identify the BSS model with lowest test error and the corresponding test error:

```
> best_bss_fit_err = min(bss_err)
> c(which.min(bss_err), best_bss_fit_err)
[1] 5.00000000 0.03221083
```

Thus, we select the BSS model with 5 predictors, with a test error ≈ 0.03221083. We fit a model over the full dataset with these predictors and summarise it:

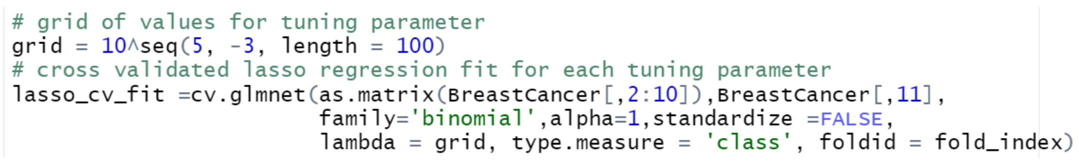
```
bss_fit$Subsets[6, ] # since this table starts from NULL model
best_bss_fit = glm(Class ~ .-Cell.size-Cell.shape-Epith.c.size-Mitoses, data = BreastCancer[, 2:11], family = 'binomial')
summary(best_bss_fit)
```

```
Call:
glm(formula = Class ~ . - Cell.size - Cell.shape - Epith.c.size - 
    Mitoses, family = "binomial", data = BreastCancer[, 2:11])

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-3.8603  -0.1228  -0.0534   0.0227   2.1903  

Coefficients:
                Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -1.2700     0.2825  -4.495 6.97e-06 ***
Cl.thickness      2.0910     0.3720   5.621 1.90e-08 ***
Marg.adhesion     1.1319     0.3321   3.409 0.000652 ***
Bare.nuclei       1.6300     0.3206   5.085 3.68e-07 ***
Bl.cromatin       1.3544     0.3679   3.681 0.000232 ***
Normal.nucleoli   1.0202     0.2986   3.417 0.000634 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We see that our selected BSS model has 5 predictors with coefficients given above, dropping *Cell.size, Cell.shape, Epith.c.size* and *Mitoses*. For *Cell.size, Cell.shape and Epith.c.size,* this seems logical from our earlier inference of the scatterplot matrix; all three predictors had reasonable correlations with other predictors indicating a lack of need to include them in the final model. In the case of *Mitoses,* it had a low sample mean and variance, indicating that there may just be very little correlation between it and the classification of a sample. Our remaining predictors are all highly significant. We see that an increase in any predictor results in an increased probability that a sample will be classed as *malignant.*

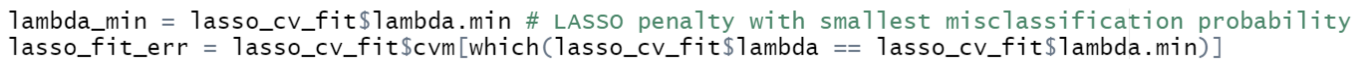
Regularised Logistic Regression with LASSO Penalties

We utilise the *glmnet* R library to perform logistic regression with a LASSO penalty on our data. Specifically we use *cv.glmnet* to generate 100 LASSO regularised logistic regression models, each with a different LASSO penalty between values 10^{-3} and 10^{5} inclusive, and each model making use of 10-fold cross validation to generate a corresponding misclassification probability using our previous fold indices.

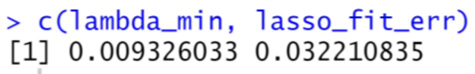


```
# grid of values for tuning parameter
grid = 10^seq(5, -3, length = 100)
# cross validated lasso regression fit for each tuning parameter
lasso_cv_fit =cv.glmnet(as.matrix(BreastCancer[,2:10]),BreastCancer[,11],
                        family='binomial',alpha=1,standardize =FALSE,
                        lambda = grid, type.measure = 'class', foldid = fold_index)
```

We identify the LASSO penalty that returns the lowest test error and the corresponding test error:

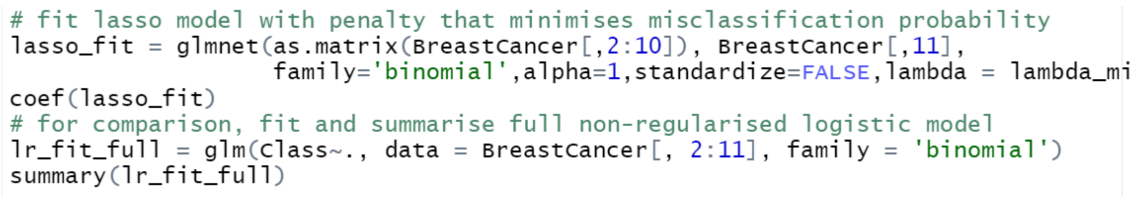


```
lambda_min = lasso_cv_fit$lambda.min # LASSO penalty with smallest misclassification probability
lasso_fit_err = lasso_cv_fit$cvm[which(lasso_cv_fit$lambda == lasso_cv_fit$lambda.min)]
```

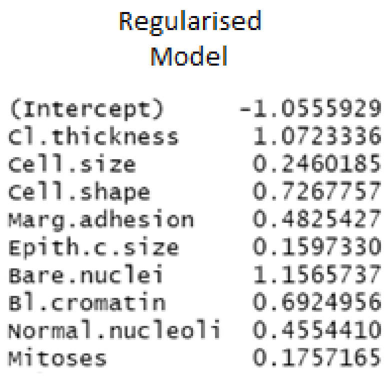


```
> c(lambda_min, lasso_fit_err)
[1] 0.009326033 0.032210835
```

Thus, we have a LASSO penalty logistic regression model with LASSO penalty ≈ 0.009326033, with test error ≈ 0.032210835. We then fit this penalised logistic regression model over the full dataset and summarise it alongside the full non-regularised model:

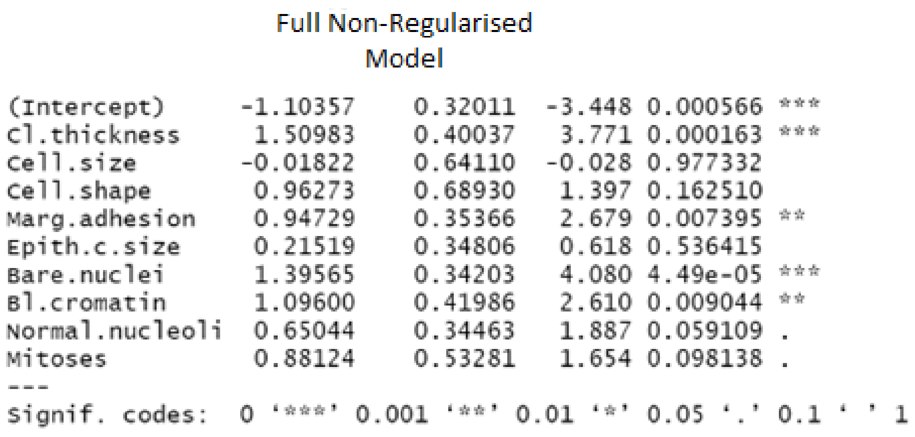


```
# fit lasso model with penalty that minimises misclassification probability
lasso_fit = glmnet(as.matrix(BreastCancer[,2:10]), BreastCancer[,11],
                   family='binomial',alpha=1,standardize=FALSE,lambda = lambda_min)
coef(lasso_fit)
# for comparison, fit and summarise full non-regularised logistic model
lr_fit_full = glm(Class~., data = BreastCancer[, 2:11], family = 'binomial')
summary(lr_fit_full)
```



Regularised
Model

```
(Intercept)     -1.0555929
Cl.thickness     1.0723336
Cell.size        0.2460185
Cell.shape       0.7267757
Marg.adhesion    0.4825427
Epith.c.size     0.1597330
Bare.nuclei      1.1565737
Bl.cromatin      0.6924956
Normal.nucleoli  0.4554410
Mitoses          0.1757165
```



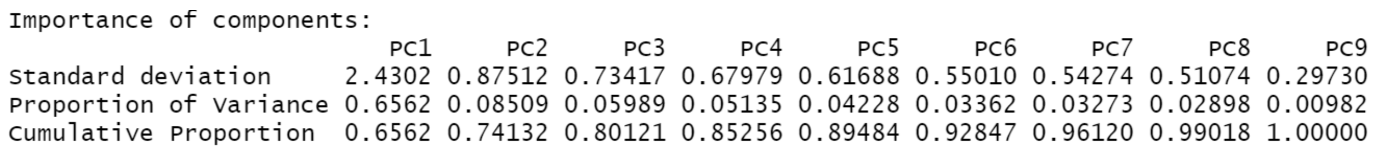
Full Non-Regularised
Model

```
(Intercept)     -1.10357   0.32011  -3.448 0.000566 ***
Cl.thickness     1.50983   0.40037   3.771 0.000163 ***
Cell.size       -0.01822   0.64110  -0.028 0.977332
Cell.shape       0.96273   0.68930   1.397 0.162510
Marg.adhesion    0.94729   0.35366   2.679 0.007395 **
Epith.c.size     0.21519   0.34806   0.618 0.536415
Bare.nuclei      1.39565   0.34203   4.080 4.49e-05 ***
Bl.cromatin      1.09600   0.41986   2.610 0.009044 **
Normal.nucleoli  0.65044   0.34463   1.887 0.059109 .
Mitoses          0.88124   0.53281   1.654 0.098138 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

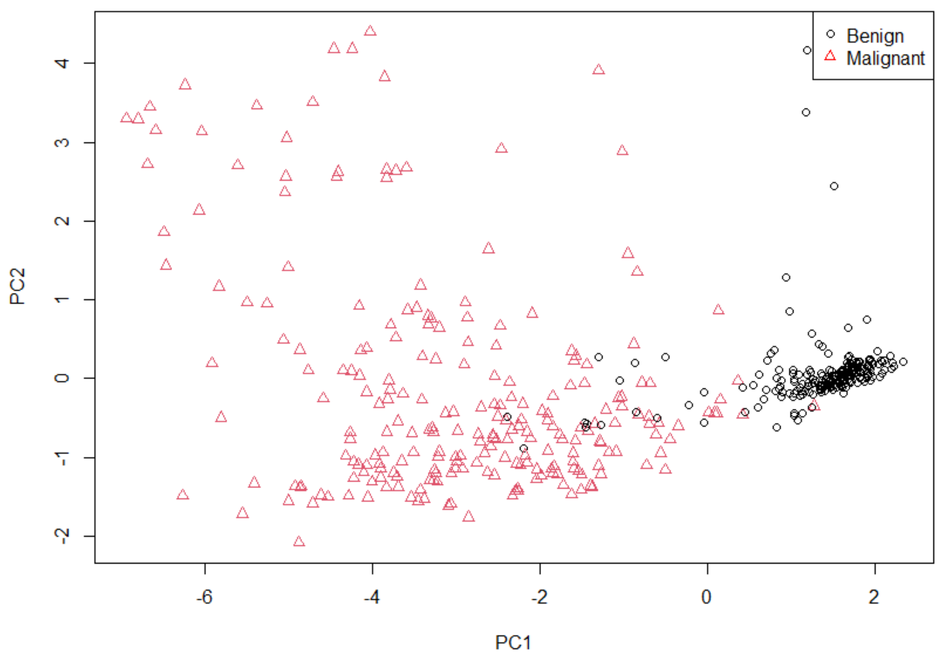
We can see that a degree of regularisation has happened for all predictors, shrinking most predictor coefficients between 0.05 and 0.5, although strangely *Cell.size* has even had its sign changed to positive and its absolute value increased but this negative effect within the non-regularised model is likely a consequence of its collinearity with *Cell.shape*. Interestingly, none of the predictors have been outright dropped from the model, although *Cell.size*, *Epith.c.size* and *Mitoses* have a small effect on classification. An increase in any of the predictors results in an increased probability for a tissue sample to be classified *malignant*.

Discriminant Analysis

We begin by summarising and plotting the first two principle components for the data:



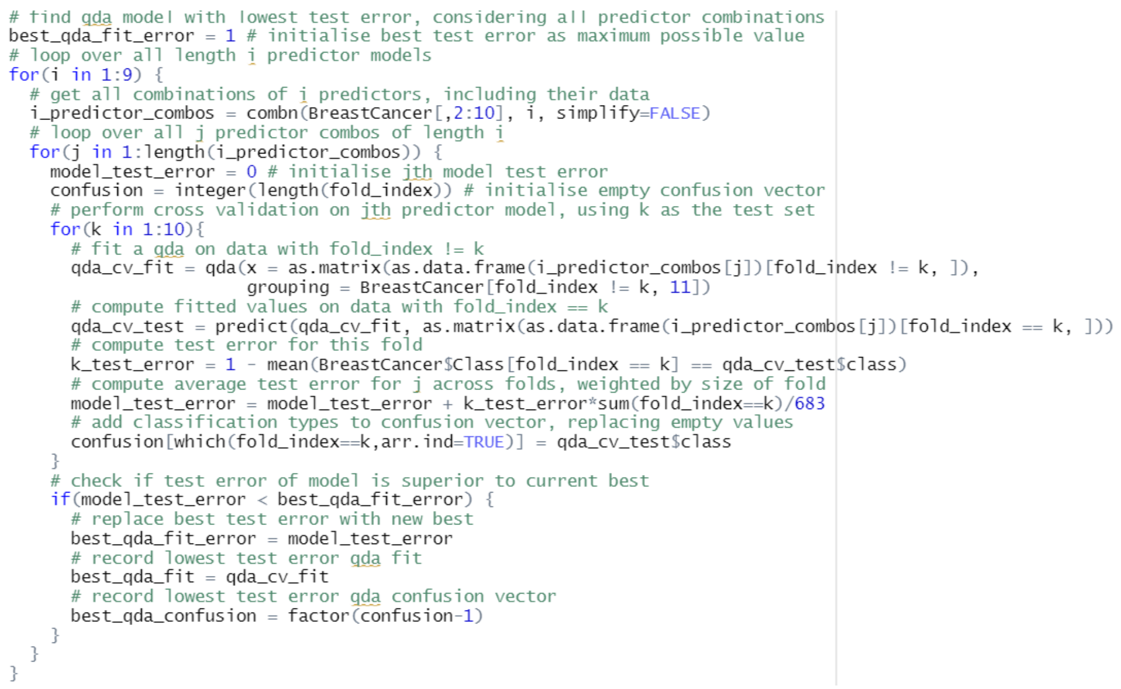
```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     2.4302 0.87512 0.73417 0.67979 0.61688 0.55010 0.54274 0.51074 0.29730
Proportion of Variance 0.6562 0.08509 0.05989 0.05135 0.04228 0.03362 0.03273 0.02898 0.00982
Cumulative Proportion  0.6562 0.74132 0.80121 0.85256 0.89484 0.92847 0.96120 0.99018 1.00000
```



We see from our summary that the cumulative proportion of variance explained by the first two principle components is ≈ 74%, and therefore use of only the first two PCs should allow us to make reasonably accurate inferences about the group variations.

It is clear from this plot that the groups differ substantially in variance. We will therefore employ a QDA to make use of its flexibility in assuming different covariance matrices. It's also worth noting that the assumption of normality within the groups does seem to be violated to some degree, especially within the *malignant* class. Nonetheless we will still pursue discriminant analysis as it can perform well under slight violations of its assumptions.

We utilise the *qda* function within the *MASS* library to generate our models using QDA. The function is run for all possible combinations of predictors, additionally computing a weighted average cross validated test error for each model with our previously defined fold indices, as well as its classifications. The model with the smallest test error is recorded alongside its error and classifications. We also assume that the prior probabilities equal the group proportions through our assumption that the patients within the dataset are a random sample from the population of women experiencing symptoms of breast cancer.



```
# find qda model with lowest test error, considering all predictor combinations
best_qda_fit_error = 1 # initialise best test error as maximum possible value
# loop over all length i predictor models
for(i in 1:9) {
  # get all combinations of i predictors, including their data
  i_predictor_combos = combn(BreastCancer[,2:10], i, simplify=FALSE)
  # loop over all j predictor combos of length i
  for(j in 1:length(i_predictor_combos)) {
    model_test_error = 0 # initialise jth model test error
    confusion = integer(length(fold_index)) # initialise empty confusion vector
    # perform cross validation on jth predictor model, using k as the test set
    for(k in 1:10){
      # fit a qda on data with fold_index != k
      qda_cv_fit = qda(x = as.matrix(as.data.frame(i_predictor_combos[j])[fold_index != k, ]),
                       grouping = BreastCancer[fold_index != k, 11])
      # compute fitted values on data with fold_index == k
      qda_cv_test = predict(qda_cv_fit, as.matrix(as.data.frame(i_predictor_combos[j])[fold_index == k, ]))
      # compute test error for this fold
      k_test_error = 1 - mean(BreastCancer$Class[fold_index == k] == qda_cv_test$class)
      # compute average test error for j across folds, weighted by size of fold
      model_test_error = model_test_error + k_test_error*sum(fold_index==k)/683
      # add classification types to confusion vector, replacing empty values
      confusion[which(fold_index==k,arr.ind=TRUE)] = qda_cv_test$class
    }
    # check if test error of model is superior to current best
    if(model_test_error < best_qda_fit_error) {
      # replace best test error with new best
      best_qda_fit_error = model_test_error
      # record lowest test error qda fit
      best_qda_fit = qda_cv_fit
      # record lowest test error qda confusion vector
      best_qda_confusion = factor(confusion-1)
    }
  }
}
```

After running the above we identify our lowest test error QDA model predictors and its test error:

```
> c(colnames(best_qda_fit$means), best_qda_fit_error)
[1] "Cl.thickness"       "Cell.shape"         "Epith.c.size"       "Bare.nuclei"        "0.0336749633967789"
```

We see that our lowest test error QDA model includes predictors *Cl.thickness, Cell.shape, Epith.c.size* and *Bare.nuclei*. It has a corresponding classification error ≈ 0.0336749633967789. We then fit this QDA model over the full dataset and summarise it:

```
# fit lowest classification error qda model and summarise
qda_fit = qda(Class ~ Cl.thickness + Cell.shape + Epith.c.size + Bare.nuclei, data = BreastCancer[, 2:11])
qda_fit
```

```
Call:
qda(Class ~ Cl.thickness + Cell.shape + Epith.c.size + Bare.nuclei,
    data = BreastCancer[, 2:11])

Prior probabilities of groups:
        0         1 
0.6500732 0.3499268 

Group means:
  Cl.thickness Cell.shape Epith.c.size Bare.nuclei
0   -0.5240440 -0.6025644   -0.5065718  -0.6031546
1    0.9735377  1.1194084    0.9410791   1.1205047
```

We have dropped 5 of our predictors in this model while attaining high prediction accuracy, evidenced through our low test error. All predictors have a roughly equal weighting in how they affect the classification of a tissue sample. As with the other models, an increased value in any predictor results in an increased probability for a sample to be classed as *malignant*, as indicated by the group means.

Selection of a Classification Model

We briefly summarise our three models in the following table:

	BSS Logistic Regression	*LASSO Logistic Regression*	*QDA*
Test Error (4 dp)	0.0322	0.0322	0.0337
Predictors	5	9	4

We first recall that each of these models were built using 10-fold cross validation on the same test error type, with the folds distributed identically for each. Therefore, the differences in test error here are subject purely to predictive power in the models. We see that the BSS and LASSO models have the lowest test error, both identical to 4 decimal places. Despite this, the test error of the QDA model is only approximately 5% larger than for the regression models, therefore the choice of predictors in the models will be the most important factor in deciding on a final model. In this regard, we will select the QDA model as our final model for classifying tissue samples due to its minimal predictors included, which is a desirable property in any statistical model for reasons such as to reduce the potential for being overfitted.

Final Classification Model – Quadratic Discriminant Analysis

We will look at the QDA model in more depth as it is our final classification model for this dataset. We note that it has a misclassification probability ≈ 0.0337 when used with 10-fold cross validation and thus we would expect it to misclassify roughly one in thirty tissue samples. We can look at the confusion matrix for the cross validated model to see the nature of its misclassifications:

```
> (confusion = table(Observed=BreastCancer[, 11], Predicted = best_qda_confusion))

        Predicted
Observed   0   1
       0 427  17
       1   6 233
```

Recalling that 0 corresponds to *Benign* and 1 corresponds to *Malignant*, we see that misclassifications by our model are generally false-positives, occuring almost three times as often as false-negatives. As the dataset is within a medical context, misclassification behaviour tending in this direction is less serious than the converse.

Returning to the variant of the model fitted on the full dataset, we remind ourselves of the predictors in this model and their means:

```
Group means:
  Cl.thickness Cell.shape Epith.c.size Bare.nuclei
0   -0.5240440 -0.6025644   -0.5065718  -0.6031546
1    0.9735377  1.1194084    0.9410791   1.1205047
```

Reiterating what was said before, our model only makes use of *Cl.thickness*, *Cell.shape*, *Epith.c.size* and *Bare.nuclei*. We can therefore say that, out of the predictors described in the dataset, these are the most important factors in deducing if a sample is cancerous. This could be that they are far stronger indicators of cancer than the other predictors or that the unincluded predictors are strongly correlated with these four predictors, the latter possibility seemed apparent in our exploratory analysis. For whichever metrics were used in the study the data in these predictors were derived from, higher values in these metrics indicate a stronger likelihood a sample will be found cancerous, with roughly equal weighting per predictor.

1.2 Theoretical Part

We have a classification rule

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

Since Y can only take values 0 or 1, this classification rule is equivalent to

$$Y = \begin{cases} 1, & \text{if } \mathrm{P}(Y = 1|\underline{X} = \underline{x}) > \alpha, \\ 0, & \text{if } 1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x}) \geq \alpha, \end{cases}$$

noting that

$$1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x}) = \mathrm{P}(Y = 0|\underline{X} = \underline{x})$$

in this case.

We can equivalently state this classification rule in terms of the odds that $Y = 1$, that is,

$$\frac{\mathrm{P}(Y = 1|\underline{X} = \underline{x})}{1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x})}.$$

For a logistic regression model, this is

$$\frac{\mathrm{P}(Y = 1|\underline{X} = \underline{x})}{1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x})} = \frac{e^{\beta_0 + \underline{\beta}_1^T \underline{x}}}{1 + e^{\beta_0 + \underline{\beta}_1^T \underline{x}}} \left(\frac{1}{1 + e^{\beta_0 + \underline{\beta}_1^T \underline{x}}} \right)^{-1}$$
$$\Rightarrow \log \left(\frac{\mathrm{P}(Y = 1|\underline{X} = \underline{x})}{1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x})} \right) = \beta_0 + \underline{\beta}_1^T \underline{x},$$

that is, the log-odds of $Y = 1$ for a logistic regression model with this classification rule form a linear boundary.

The odds for a discriminant rule to classify $Y = 1$ on a partition formed of $K = 2$ regions are, again noting that

$$1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x}) = \mathrm{P}(Y = 0|\underline{X} = \underline{x})$$

in this case,

$$\frac{\mathrm{P}(Y = 1|\underline{X} = \underline{x})}{1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x})} = \frac{f_1(\underline{x})\pi_1}{\sum_{i=0}^{1} f_i(\underline{x})\pi_i} \times \frac{\sum_{i=0}^{1} f_i(\underline{x})\pi_i}{f_0(\underline{x})\pi_0}$$
$$= \frac{f_1(\underline{x})\pi_1}{f_0(\underline{x})\pi_0}.$$

Within a linear discriminant analysis, a special case of this, we assume that $\underline{X}|Y$ is multivariate normal with K mean vectors $\underline{\mu}_i,\ i = 0, 1, \ldots, K-1$ and a single shared covariance matrix Σ. Thus our classification odds for $Y = 1$ on p predictors are

$$\frac{\mathrm{P}(Y = 1|\underline{X} = \underline{x})}{1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x})} = \frac{(2\pi)^{-p/2}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(\underline{x} - \underline{\mu}_1)^{\mathrm{T}}\Sigma^{-1}(\underline{x} - \underline{\mu}_1)\right\}\pi_1}{(2\pi)^{-p/2}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(\underline{x} - \underline{\mu}_0)^{\mathrm{T}}\Sigma^{-1}(\underline{x} - \underline{\mu}_0)\right\}\pi_0}$$
$$= \frac{\pi_1}{\pi_0}\exp\left\{\frac{1}{2}(\underline{x} - \underline{\mu}_0)^{\mathrm{T}}\Sigma^{-1}(\underline{x} - \underline{\mu}_0) - \frac{1}{2}(\underline{x} - \underline{\mu}_1)^{\mathrm{T}}\Sigma^{-1}(\underline{x} - \underline{\mu}_1)\right\}$$
$$\Rightarrow \log\left(\frac{\mathrm{P}(Y = 1|\underline{X} = \underline{x})}{1 - \mathrm{P}(Y = 1|\underline{X} = \underline{x})}\right)$$

$$
\begin{aligned}
&= \log\frac{\pi_1}{\pi_0} + \frac{1}{2}(\underline{x} - \underline{\mu}_0)^{\mathrm{T}}\Sigma^{-1}(\underline{x} - \underline{\mu}_0) - \frac{1}{2}(\underline{x} - \underline{\mu}_1)^{\mathrm{T}}\Sigma^{-1}(\underline{x} - \underline{\mu}_1) \\
&= \log\frac{\pi_1}{\pi_0} - \frac{1}{2}(\underline{\mu}_1 + \underline{\mu}_0)^{\mathrm{T}}\Sigma^{-1}(\underline{\mu}_1 - \underline{\mu}_0) + \underline{x}^{\mathrm{T}}\Sigma^{-1}(\underline{\mu}_1 - \underline{\mu}_0) \\
&= \alpha_0 + \underline{\alpha}_1^{\mathrm{T}}\underline{x},
\end{aligned}
$$

where we have defined

$$
\alpha_0 = \log\frac{\pi_1}{\pi_0} - \frac{1}{2}(\underline{\mu}_1 + \underline{\mu}_0)^{\mathrm{T}}\Sigma^{-1}(\underline{\mu}_1 - \underline{\mu}_0),
$$
$$
\underline{\alpha}_1 = \Sigma^{-1}(\underline{\mu}_1 - \underline{\mu}_0).
$$

Thus, in the case of a linear discriminant analysis on $K = 2$ regions, the log-odds of $Y = 1$ form a linear boundary.

Thus, we have that logistic regression with the given classification rule and a discriminant rule over two regions with a linear boundary between these regions take precisely the same form in their log-odds; they must therefore be equivalent.

Appendix

In this appendix, the full R code used for the analysis is referenced.

```r
library(mlbench)
library(bestglm)
library(glmnet)
library(MASS)
data(BreastCancer)

dim(BreastCancer)
head(BreastCancer)



# data cleaning


# convert cytological characteristics to quantitative variables
BreastCancer[, 2:10] = lapply(BreastCancer[, 2:10], as.numeric)
# check conversion has been applied successfully
lapply(BreastCancer, class)
# remove rows containing missing values
BreastCancer = na.omit(BreastCancer)



# exploratory data analysis


# count of each response factor
table(BreastCancer$Class)

# scatterplot matrix of predictors, coloured by response factor, noise added
pairs(lapply(BreastCancer[, 2:10], jitter), col = BreastCancer[, 11])

# compute sample mean vector of predictors
colMeans(BreastCancer[, 2:10])
# compute sample variances of predictors
diag(var(BreastCancer[, 2:10]))
# compute sample correlation matrix of predictors
cor(BreastCancer[, 2:10])



# classification modelling


# standardise data
BreastCancer[, 2:10] = scale(BreastCancer[, 2:10])

# code Class as binary
BreastCancer[, 11] = as.numeric(BreastCancer[, 11])-1

# ensure reproducibility and fairness in model comparison
set.seed(1)

# cross validation folds, used for all models
fold_index = sample(rep(1:10, length = 683)) # n = 683


# non-regularised logistic regression with bss


# bss algorithm on 9 predictors
bss_fit = bestglm(BreastCancer[, 2:11], family = binomial)
bss_fit$Subsets

# compute test error for BSS models
# primary function, uses k-fold validation to find misclassification probability for a model
reg_cv = function(X1, y, fold_ind) {
  Xy = data.frame(X1, y=y) # get data frame formed of predictors and response
  nfolds = max(fold_ind) # get total number of folds
  if(!all.equal(sort(unique(fold_ind)), 1:nfolds)) stop("Invalid fold partition.")
  cv_errors = numeric(nfolds) # empty vector of cv errors
  for(fold in 1:nfolds) {
    tmp_fit = glm(y ~ ., data=Xy[fold_ind!=fold,], family = 'binomial') # fit predictive model
    yhat = predict(tmp_fit, Xy[fold_ind==fold,], type = 'response') # get predicted probs
    yhat_class = round(yhat) # round predicted probs to 1 or 0, i.e. get their classification
    yobs = y[fold_ind==fold] # find the actual classifications in the data set
    cv_errors[fold] = 1 - mean(yobs == yhat_class) # get misclassification probability overall
  }
  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) # weighted by fold sizes, getting overall error
  return(test_error)
}

# pass each BSS model into the above primary function, returning error for each
reg_bss_cv = function(X1, y, best_models, fold_index) {
  p = ncol(X1)
  test_errors = numeric(p)
  for(k in 1:p) { # pass each candidate model into the above function
    test_errors[k] = reg_cv(X1[,best_models[k,]], y, fold_index)
  }
  return(test_errors)
}

# use above functions to return error of each BSS model
bss_err = reg_bss_cv(BreastCancer[, 2:10], BreastCancer[, 11],
                     as.matrix(bss_fit$Subsets[-1,2:10]), fold_index)

# identify lowest test error model and its error
best_bss_fit_err = min(bss_err)
c(which.min(bss_err), best_bss_fit_err)

# view model with lowest error
bss_fit$Subsets[6, ] # since this table starts from NULL model
best_bss_fit = glm(Class ~ .-Cell.size-Cell.shape-Epith.c.size-Mitoses, data = BreastCancer[, 2:11], family =
'binomial')
summary(best_bss_fit)


# regularised logistic regression with lasso


# grid of values for tuning parameter
grid = 10^seq(5, -3, length = 100)
# cross validated lasso regression fit for each tuning parameter
lasso_cv_fit =cv.glmnet(as.matrix(BreastCancer[,2:10]),BreastCancer[,11],
                        family='binomial',alpha=1,standardize =FALSE,
                        lambda = grid, type.measure = 'class', foldid = fold_index)

# extract LASSO penalty with smallest misclassification probability, and corresponding probability
lambda_min = lasso_cv_fit$lambda.min # LASSO penalty with smallest misclassification probability
lasso_fit_err = lasso_cv_fit$cvm[which(lasso_cv_fit$lambda == lasso_cv_fit$lambda.min)]
c(lambda_min, lasso_fit_err)

# fit lasso model with penalty that minimises misclassification probability
lasso_fit = glmnet(as.matrix(BreastCancer[,2:10]), BreastCancer[,11],
                   family='binomial',alpha=1,standardize=FALSE,lambda = lambda_min)
coef(lasso_fit)
# for comparison, fit and summarise full non-regularised logistic model
lr_fit_full = glm(Class~., data = BreastCancer[, 2:11], family = 'binomial')
summary(lr_fit_full)
```

```
# discriminant analysis


# derive and plot first two principle components of data

pca = prcomp(BreastCancer[, 2:10]) # pca of predictors
summary(pca) # find variance explained by first 2 predictors
# Plot the first PC against the second PC with colours and characters in 'Class'
plot(pca$x[,1], pca$x[,2], xlab='PC1', ylab='PC2', col= BreastCancer$Class+1, pch= BreastCancer$Class+1)
legend(x = 'topright', legend = c('Benign','Malignant'), col= c('black','red'), pch = c(1,2))

# find qda model with lowest test error, considering all predictor combinations
best_qda_fit_error = 1 # initialise best test error as maximum possible value
# loop over all length i predictor models
for(i in 1:9) {
  # get all combinations of i predictors, including their data
  i_predictor_combos = combn(BreastCancer[,2:10], i, simplify=FALSE)
  # loop over all j predictor combos of length i
  for(j in 1:length(i_predictor_combos)) {
    model_test_error = 0 # initialise jth model test error
    confusion = integer(length(fold_index)) # initialise empty confusion vector
    # perform cross validation on jth predictor model, using k as the test set
    for(k in 1:10){
      # fit a qda on data with fold_index != k
      qda_cv_fit = qda(x = as.matrix(as.data.frame(i_predictor_combos[j])[fold_index != k, ]),
                       grouping = BreastCancer[fold_index != k, 11])
      # compute fitted values on data with fold_index == k
      qda_cv_test = predict(qda_cv_fit, as.matrix(as.data.frame(i_predictor_combos[j])[fold_index == k, ]))
      # compute test error for this fold
      k_test_error = 1 - mean(BreastCancer$Class[fold_index == k] == qda_cv_test$class)
      # compute average test error for j across folds, weighted by size of fold
      model_test_error = model_test_error + k_test_error*sum(fold_index==k)/683
      # add classification types to confusion vector, replacing empty values
      confusion[which(fold_index==k,arr.ind=TRUE)] = qda_cv_test$class
    }
    # check if test error of model is superior to current best
    if(model_test_error < best_qda_fit_error) {
      # replace best test error with new best
      best_qda_fit_error = model_test_error
      # record lowest test error qda fit
      best_qda_fit = qda_cv_fit
      # record lowest test error qda confusion vector
      best_qda_confusion = factor(confusion-1)
    }
  }
}
# identify lowest test error model predictors and test error
c(colnames(best_qda_fit$means), best_qda_fit_error)

# fit lowest classification error qda model and summarise
qda_fit = qda(Class ~ Cl.thickness + Cell.shape + Epith.c.size + Bare.nuclei, data = BreastCancer[, 2:11])
qda_fit



# final classification model summaries


# confusion matrix of cv qda model
(confusion = table(Observed=BreastCancer[, 11], Predicted = best_qda_confusion))

# summary of qda model
qda_fit
```