MATH42715: ISDS Assignment 2 Report

Measuring the extent to which a tissue sample can be classified as benign or malignant using nine cytological characteristics

Anonymous Marking Code: Z0182576 Date: December 2022

1 Introduction

This report aims to measure the extent to which a tissue sample could be classified as benign or malignant using nine cytological characteristics. These nine characteristics are: Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, and Mitoses. Some 699 samples have been studied and stored in the Wisconsin Breast Cancer Database, from which we will perform suitable exploratory data analysis and use it to develop classification models. The samples are randomly taken from the population of women experiencing symptoms of breast cancer. For each of the samples, each of these characteristics were given a score from 1 to 10, where smaller numbers represent healthier looking cells. We then train multiple classification algorithms using 10-fold cross validation to determine which classification algorithm presents the lowest test error (i.e. has the lowest misclassification rate), and will thus be most accurate at classifying a new sample as either benign or malignant.

2 Cleaning the Data

To begin, we note that alongside the first "Id" character variable (which for the purposes of this study we will ignore), the nine characteristics are stored as factors, and to perform meaningful exploratory data analysis we must first convert these to quantitative variables. This can be easily done using a for loop as follows:

```
for (i in 2:10){
  BreastCancer[,i] = as.integer(BreastCancer[,i])
}
```

We check that the conversion to quantitative variables has taken place using `is.factor(BreastCancer $Cell.size)`, which returns `FALSE`. We now move onto the next step of cleaning the data.

We continue by omitting any rows with incomplete data. First, we check how many instances of `NA` are present using `sum(is.na(BreastCancer))`, which returns 16. These rows are then excluded from the data set using `BreastCancer = (na.omit(BreastCancer))`. Then, `sum(is.na(BreastCancer))` is called again, this time returning 0, indicating that all the incomplete records have been removed.

3 Exploratory Data Analysis

Now we can begin exploring the cleaned data. We set `n = nrow(BreastCancer)` and `p = ncol(BreastCancer)-2`, giving us `n = 683` complete observations and `p = 9` predictor variables corresponding to the 9 quantitative cytological characteristics. Since we only have two classes, benign or malignant, we will take the number of groups `K = 2`. This will allow us to perform a logistic regression. We note the frequencies of each class in the dataset:

```
> table(BreastCancer$Class)
   benign malignant
      444       239
```

Since these are random samples from the population of women experiencing symptoms of breast cancer, we can assume that the probability of having a malignant tumour for any woman experiencing breast cancer symptoms can be estimated by $\frac{239}{683}$. This will serve as the prior group membership probability π_{mal} which will be useful in creating the Bayes classifer for both linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). Naturally, we will take the prior group membership probability π_{ben} for benign tumours as $1 - \frac{239}{683} = \frac{444}{683}$.

We want to see how these benign and malignant tumour cases are distributed among the data. We will do this using a scatterplot matrix, wherein we will be able to see the relationship between all the predictor variables and the response variable. First, we will also change the entries for the "Class" variable in the `BreastCancer` dataset into integers 0 and 1, so that they can be used as colour arguments for black and red in the scatterplot matrix, and thus we will be able to distinguish the benign and malignant samples at a glance. To do this, we check the values of the `Class` variable:

```
> head(as.integer(BreastCancer$Class))
[1] 1 1 1 1 1 2
```

Here, 1 denotes a benign tumour while 2 denotes a malignant one. Combining the effort to convert these into "0" and "1" values and reduce unimportant data, we will introduce another dataframe `BC` which only contains the 9 predictor variables and the newly transformed `Class` variable, where Class only takes the values 0 (benign) and 1 (malignant), i.e. we remove the "Id" class.

```
BC = data.frame(BreastCancer[,2:10],Class=as.integer(BreastCancer$Class)-1)
```

Now we produce a scatterplot matrix using `pairs(BC[1:9], col = BC[,10]+1, pch = BC[,10])`, which will colour the benign samples black and the malignant samples red: There are a couple obvious trends.

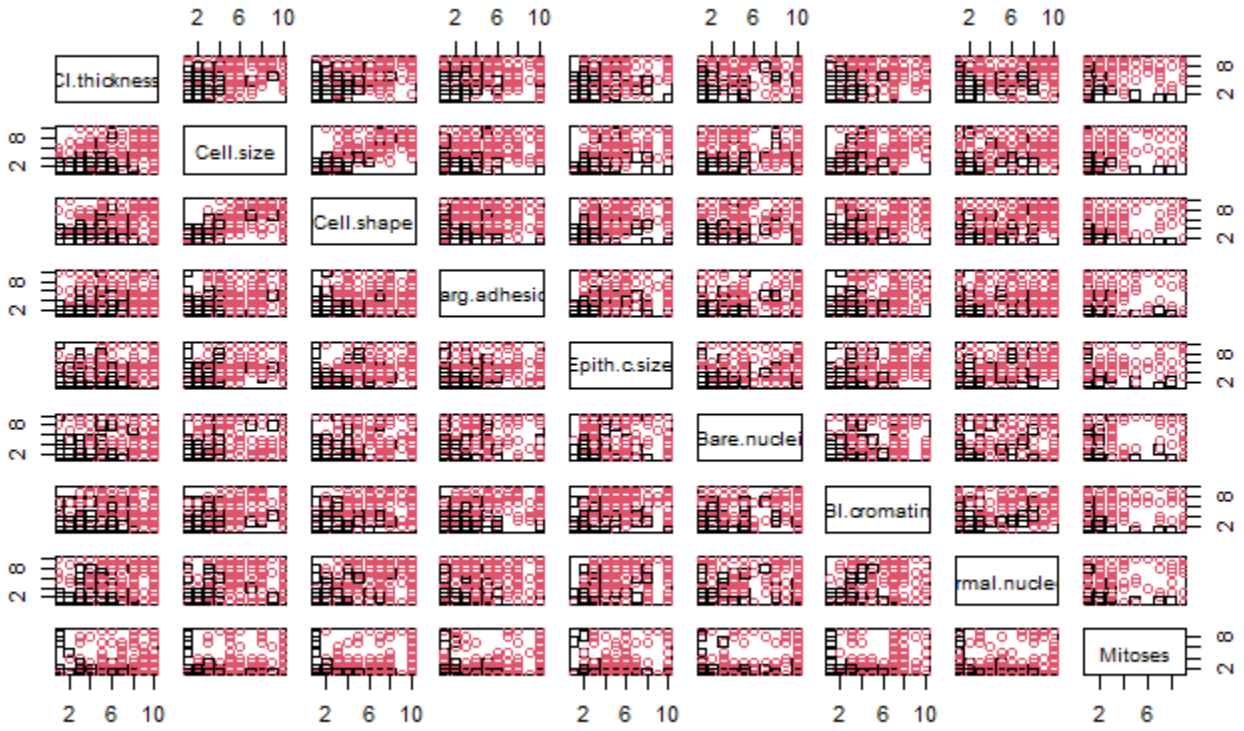

Figure 1: Scatterplots showing relationship between predictor variables. Black: benign. Red: malignant.

Firstly, we note that the benign samples tend to be clustered in the bottom left of each plot, which is associated with low values in both variables. This is in concordance with the idea that lower numbers indicated healthier looking cells. The malignant samples seem to be distributed everywhere however, even for samples with low values for its variables. Notably, the top right of each plot is dominated by red points. In other words, given that we choose a sample with high values in the predictor variables, there is a very high probability that it is a malignant tumour.
The second trend is the strong positive linear association between uniformity of cell size and uniformity of cell shape: generally, as the value for one increases, so does the other. Their notable correlation could mean that it would be redundant to use both the `Cell.size` and `Cell.shape` variables, and only one may be necessary to prevent too many predictor variables hurting the accuracy of the model we will fit. On the other hand, we see that there aren't many other obvious relationships between predictor variables.
For coverage, we will also look at another way to assess the relationship between variables. Recall that the points for the malignant samples were distributed across the entire space in the scatterplot matrix. This made it hard to see a correlation between any predictor variable and the response variable. Now, we will use the `cor()` function to produce a correlation matrix of Pearson's correlation coefficient between every two variables in our data frame `BC`.

```
cor(BC[1:10])
```

	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.nuclei	Bl.cromatin	Normal.nucleoli	Mitoses	Class
Cl.thickness	1.0000000	0.6424815	0.6534700	0.4878287	0.5235960	0.5930914	0.5537424	0.5340659	0.3545301	0.7147899
Cell.size	0.6424815	1.0000000	0.9072282	0.7069770	0.7535440	0.6917088	0.7555592	0.7193460	0.4654091	0.8208014
Cell.shape	0.6534700	0.9072282	1.0000000	0.6859481	0.7224624	0.7138775	0.7353435	0.7179634	0.4468571	0.8218909
Marg.adhesion	0.4878287	0.7069770	0.6859481	1.0000000	0.5945478	0.6706483	0.6685671	0.6031211	0.4249917	0.7062941
Epith.c.size	0.5235960	0.7535440	0.7224624	0.5945478	1.0000000	0.5857161	0.6181279	0.6289264	0.4811836	0.6909582
Bare.nuclei	0.5930914	0.6917088	0.7138775	0.6706483	0.5857161	1.0000000	0.6806149	0.5842802	0.3490108	0.8226959
Bl.cromatin	0.5537424	0.7555592	0.7353435	0.6685671	0.6181279	0.6806149	1.0000000	0.6656015	0.3536683	0.7582276
Normal.nucleoli	0.5340659	0.7193460	0.7179634	0.6031211	0.6289264	0.5842802	0.6656015	1.0000000	0.4370424	0.7186772
Mitoses	0.3545301	0.4654091	0.4468571	0.4249917	0.4811836	0.3490108	0.3536683	0.4370424	1.0000000	0.4312971
Class	0.7147899	0.8208014	0.8218909	0.7062941	0.6909582	0.8226959	0.7582276	0.7186772	0.4312971	1.0000000

Figure 2: Correlation matrix of variables

This table shows us that the strongest correlated variable with `Class` is `Bare.nuclei`, with a correlation of 0.8226959 followed by `Cell.size` and `Cell.shape`. This would mean that samples with high `Bare.nuclei`, `Cell.size` and `Cell.shape` scores are strongly associated with being malignant. Looking back at the scatterplots that involve these variables, we can easily identify the change in distribution from black and red spots to majority red spots. In fact, the correlation matrix indicates that all the predictor variables are positively correlated with `Class`, even the weakest correlated variable `Mitoses` which has a correlation of 0.4312971. We confirm that the `Cell.size` and `Cell.shape` variables are strongly positively correlated with a correlation of 0.9072282, and note that other pairs of predictor variables do not have as strong a correlation, as expected from the scatterplot.

4 Modelling

Now that we have developed some understanding of the data and the patterns it holds, it is appropriate to try to develop a model that will accurately classify whether new observations are benign or malignant. In this section we use a process called "Best Subset Selection" to choose the best subset of predictor variables that will allow the model to perform most accurately. Our first model will be a logistic regression model, and then we will create linear and quadratic discriminant analysis models with the chosen variables to determine which is best. In logistic regression, we train the model to be able to predict classifications by finding

'regression coefficients'. These essentially indicate how and how much each predictor variable affects the response variable. Positive coefficients for a predictor variable indicate that, should all the other variables be kept constant, increasing that predictor variable will increase the value of the response variable - and opposite for negative regression coefficients. As a preparatory exercise, we will create a "rough" model to learn about the p-values of the coefficients for each predictor variable. This will help us see which variables we should expect to exclude from our final model.

```
first_log_fit = glm(Class ~ ., data = BC, family = "binomial")
summary(first_log_fit)
```

```
...
Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)    -10.110096   1.173774  -8.613  < 2e-16 ***
Cl.thickness     0.535256   0.141938   3.771 0.000163 ***
Cell.size       -0.005943   0.209158  -0.028 0.977332
Cell.shape       0.322136   0.230644   1.397 0.162510
Marg.adhesion    0.330694   0.123462   2.679 0.007395 **
Epith.c.size     0.096797   0.156568   0.618 0.536415
Bare.nuclei      0.383015   0.093865   4.080 4.49e-05 ***
Bl.cromatin      0.447401   0.171392   2.610 0.009044 **
Normal.nucleoli  0.213074   0.112894   1.887 0.059109 .
Mitoses          0.538551   0.325615   1.654 0.098138 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
...
```

From this we learn that only the `Intercept`, `Cl.thickness`, `Marg.adhesion`, `Bare.nuclei` and `Bl.cromatin` variables have a p-value less than 0.05, and so those are the only values we can be sure are different from 0 at the 95% confidence level. The `Epith.c.size`, `Cell.size` and `Cell.shape` p-values are quite high and so it is more likely that some combination of them will be excluded from the final model. Now we aim to perform Best Subset Selection to select the best predictor variables, looking using both the "AIC" and "BIC" information criteria. The best model according to these information criteria is one which returns the lowest AIC or BIC respectively.

```
bss_AIC = bestglm(BC, family=binomial, IC="AIC")
> bss_AIC$Subsets
...
    logLikelihood   AIC
0     -442.17509 884.3502
1     -127.37980 256.7596
2      -83.15598 170.3120
3      -67.77778 141.5556
4      -61.37155 130.7431
5      -56.13177 122.2635
6      -53.57186 119.1437
7*     -51.63998 117.2800
8      -51.45031 118.9006
9      -51.44991 120.8998
...
```

AIC indicates that a 7 predictor model, which we denote by $\mathcal{M}_7$, is best.

```
Intercept Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei Bl.cromatin
TRUE           TRUE     FALSE       TRUE         TRUE        FALSE        TRUE        TRUE
Normal.nucleoli Mitoses         logLikelihood      AIC
TRUE             TRUE          -51.63998        117.2800
```

Looking at the variables selected, we find that it omits `Epith.c.size`, `Cell.size` as we might have suspected from our preliminary test. We now fit and analyse a model according to the BIC criteria.

```
bss_BIC = bestglm(BC, family=binomial, IC="BIC")
bss_BIC$Subsets
...
   logLikelihood      BIC
0     -442.17509 884.3502
1     -127.37980 261.2861
2      -83.15598 179.3649
3      -67.77778 155.1351
4      -61.37155 148.8491
5*     -56.13177 144.8960
6      -53.57186 146.3027
7      -51.63998 148.9654
8      -51.45031 155.1126
9      -51.44991 161.6383
...
Intercept Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei Bl.cromatin
TRUE         TRUE     FALSE      FALSE          TRUE        FALSE        TRUE        TRUE
Normal.nucleoli  Mitoses
TRUE              FALSE
```

The BIC criteria prefers a 5 predictor model based on `Cl. thickness`, `Marg.adhesion`, `Bare.nuclei`, `Bl.cromatin`, `Normal.nucleoli`, (which we call $\mathcal{M}_5$) in contrast to the AIC's suggested 7 predictor model. Since we aim to fit only one logistic regression model, we will perform cross validation to help determine which models present the smallest error, and then decide using that data.
We will use two functions: one called `log_reg_fold_error` and another called `ten_fold_cv_error`, both left in the appendix for the curious reader. The first function calculates the misclassification rate of a logistic regression model trained with the input data. In the second function, as the name suggests, the observations will be split into ten folds via random assignment (with the limitation that all folds must have at least one observation). All the data not in this fold is used to train the model, then the model is tested using the data in the fold. The test error, called the misclassification rate, is calculated for each of the ten folds of the data. This process is repeated for models with increasing numbers of predictor variables up to $p+1$, remembering to account for the model with 0 predictors. Both these functions are provided by Dr. Heaps at [Hea] as the `logistic_reg_fold_error` and `logistic_reg_bss_cv` functions, although the latter is slightly adapted to always be 10-fold instead of allowing a general input. The purpose of dividing into 10 folds is because the number is not too large so as to be computationally expensive, but also it satisfies a bias-variance tradeoff. This is because high k would result in excessive overlap of the training data, and so there would be high variance of results, but low k would result in smaller training sets and so more bias in results.

Calling `ten_fold_cv_error` and passing in a set of fold indices generated by `sample(10, n, replace = "TRUE")` (NB: The seed is set to 5, i.e. `set.seed(5)`), we obtain that the model with the smallest cross validation test error has 7 predictors. This would lead us to conclude that we should go with our $\mathcal{M}_7$ model as suggested via the AIC criterion. Finally, we can fit a logistic regression model using these 'best' variables.

```
BC_BSS = BC[c(1,3,4,6,7,8,9,10)]
second_log_fit = glm(Class ~ ., data = BC_BSS, family = "binomial")
summary(second_log_fit)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    -9.98954    1.12478  -8.881  < 2e-16 ***
Cl.thickness    0.53425    0.14070   3.797 0.000146 ***
Cell.shape      0.34503    0.17162   2.010 0.044393 *
Marg.adhesion   0.34261    0.11923   2.873 0.004060 **
```

```
Bare.nuclei      0.38830    0.09359   4.149 3.34e-05 ***
Bl.cromatin      0.46222    0.16820   2.748 0.005997 **
Normal.nucleoli  0.22618    0.11099   2.038 0.041561 *
Mitoses          0.53536    0.32088   1.668 0.095237 .
```

Looking at the coefficients of our fitted model we can see that all coefficients of the predictor variables are positive, and so increasing the value of any of these predictor variables will increase the probabillity that the sample is classified as malignant. `Mitoses` has the largest coefficient, 0.53536, which indicates that it is the 'strongest' variable with regards to its impact on the classification. In fitting this logistic regression model, we have assumed little collinearity between predictor variables, which holds as we inspected in our correlation matrix in Figure 2. We also assume that the samples are representative, which is justified since we know that the samples were taken randomly from women experiencing symptoms of breast cancer (and not deliberately selected to influence the group proportions).

Now we fit linear and quadratic discriminant analysis models using the same set of predictor variables. Discriminant analysis operates by dividing the space of all the possible values for $\mathbf{x} = (x_1, \ldots, x_p)$, which represents a vector of the p predictor variables for a problem, into regions called "allocation regions". These divisions are made depending on the value of special functions called "discriminant function" at $\mathbf{x}$. There is a discriminant function for each group (here, there would be discriminant functions for the benign group and one for malignant group), and the discriminant function with the highest value at $\mathbf{x}$ dictates the group to which $\mathbf{x}$ belongs.

```
> lda_fit = lda(Class ~ ., data= BC_BSS)
> lda_fit
Call:
lda(Class ~ ., data = BC_BSS)

Prior probabilities of groups:
        0         1
0.6500732 0.3499268

Group means:
  Cl.thickness Cell.shape Marg.adhesion Bare.nuclei Bl.cromatin Normal.nucleoli  Mitoses
0     2.963964   1.414414      1.346847    1.346847    2.083333        1.261261 1.065315
1     7.188285   6.560669      5.585774    7.627615    5.974895        5.857741 2.543933

> qda_fit = qda(Class ~ ., data= BC_BSS)
> qda_fit
Call:
qda(Class ~ ., data = BC_BSS)

Prior probabilities of groups:
        0         1
0.6500732 0.3499268

Group means:
  Cl.thickness Cell.shape Marg.adhesion Bare.nuclei Bl.cromatin Normal.nucleoli  Mitoses
0     2.963964   1.414414      1.346847    1.346847    2.083333        1.261261 1.065315
1     7.188285   6.560669      5.585774    7.627615    5.974895        5.857741 2.543933
```

Both discriminant analysis models calculate the prior probabilities of the benign and malignant groups in exactly the same way as we did in our exploratory data analysis ($\frac{239}{683} \approx 0.3499268 = \pi_{mal}$ for the malignant group, and $\frac{444}{683} \approx 0.6500732 = \pi_{ben}$ for the benign group). We also note the group means: for each of the benign and malignant groups, we are provided with the means of each of the predictor variables. Note that when the tumour is benign, i.e. group 0, the values of the group means are lower than the values of the group mean when the tumour is malignant, i.e. group 1, which confirms the idea that lower values mean

that the tissue sample is healthier.

Both forms of discriminant analyses assume that the groups k are multivariate normal, each with their corresponding mean vector $\boldsymbol{\mu}_k$. For the purpose of this report, we will assume that this assumption holds. In LDA, the measure of uncertainty for the distribution is explained by a covariance matrix Σ for all the groups k, whereas in QDA, the values in the covariance matrix are able to change by group, so each group has a covariance matrix Σ_k. This may result in a different test error than to LDA that we will investigate in the next section.

5 Model Comparisons

Now that we have fitted our three models, we wish to see how they perform - which is best - by calculating the misclassification rate for each model. To do this, we will use 10-fold cross validation. To keep this analysis fair, we will use the same `fold_indices` for each error calculation to keep the folds the same, and so comparing errors between models is more reliable. This is because fluctuations in error will come from differences in the models themselves rather than arising via random chance. We also use a weighted mean when calculating the fold error so as to prevent the results being skewed disproportionately by a large fold that has a larger or smaller error than another fold (since the folds can take any size, as long as they are all partially filled). To perform these error calculations, we will once again use some code provided by Dr Heaps at [Hea], which one may find in the Appendix. We will use the `ten_fold_cv`, `log_reg_fold_error`, `lda_fold_error` and the `qda_fold_error` functions, which have been slightly altered for the purposes of this report to ensure that they are more specific to this case, e.g. altered for 10-fold cross validation and not any number k.
To explain what each function does: `ten_fold_cv` performs a 10-fold cross validation as described when we were deciding on the best subset of variables to choose for our logistic regression model.
`log_reg_fold_error` calculates the error of the fold currently in the loop of the `ten_fold_cv` function. We post the code here for clarity [Hea]:

```
log_reg_fold_error = function(X, y, test_indices) {

  Xy = data.frame(X, y=y)

  if(ncol(Xy)>1) fold_fit = glm(y ~ ., data=Xy[!test_indices,], family="binomial")
  else fold_fit = glm(y ~ 1, data=Xy[!test_indices,,drop=FALSE], family="binomial")

  p_hat = predict(fold_fit, Xy[test_indices,,drop=FALSE], type="response")
  y_hat = ifelse(p_hat > 0.5, 1, 0)
  y_obs = y[test_indices]
  test_error = 1 - mean(y_obs == y_hat)
  return(test_error)
}
```

The first line creates a data frame from the matrix of predictor variables and vector of response variables provided when the function is called. The function also takes in a list of test indices, which are indices corresponding to the test data. Then, the logistic regression fit is made, and a calculation is made for every observation in the test data. These calculations are stored in `p_hat`. Using the `ifelse` function, we loop through the calculations, and depending on whether or not the calculation is greater than 0.5 we classify it as benign (less than or equal to 0.5) or malignant (greater than 0.5). These predicted classifications are stored in `y_hat`, and are compared directly with the actual classes of the observations stored in `y_obs` . The test error is then calculated as 1 (representing completely imperfect classification of every observation in the test data) take away `mean(y_obs == y_hat)` which is a vector that calculates the average success of the model. This leaves us with the average failure rate, the misclassification rate.
This function works very similarly to the `lda_fold_error` and `qda_fold_error` functions, except that the

model is trained using LDA and QDA instead of logistic regression, and we have no number α to compare a regression prediction to in order to obtain a classification - since the classifications are made using the discriminant functions and allocation regions instead.

Now we calculate the errors:

```
> log_test_error = ten_fold_cv(BC_BSS[,1:7], BC_BSS[,8], fold_indices, log_reg_fold_error)
> log_test_error
[1] 0.03221083
> lda_test_error = ten_fold_cv(BC_BSS[,1:7], BC_BSS[,8], fold_indices, lda_fold_error)
> lda_test_error
[1] 0.04099561
> qda_test_error = ten_fold_cv(BC_BSS[,1:7], BC_BSS[,8], fold_indices, qda_fold_error)
> qda_test_error
[1] 0.04831625
```

Looking at the error rates, it appears that the logistic regression model provides the smallest misclassification rate, making it the most accurate. The misclassifcation rate is calculated as 0.03221083, which corresponds to roughly 3.2%. This means that for any new observation we can expect a roughly 3.2% chance that it is classified incorrectly, i.e. as benign when it is actually malignant and vice versa. The rates for LDA and QDA respectively are 0.04099561 and 0.04831625, indicating that LDA appears to be more accurate than QDA. This may be because two separate covariance matrix is needed in QDA compared to just one in LDA, which means an extra parameter needs to be estimated in QDA. This extra parameter estimation may introduce more variance, while the information gain from the extra parameter may not have been enough to negate this increase in variance - perhaps there is no reason to assume that the covariance matrix varies between the benign and malignant groups at all. Thus we see an increase the test error rate, as visible in the higher misclassification rate for QDA.

6 Conclusion

Overall, we observed that the logistic regression model had the lowest misclassification rate, followed by the LDA model and then the QDA model, so the logistic regression model is the most accurate, and the QDA model is the least accurate - not significantly so, but notably so. Based on this data, doctors and scientists should choose a logistic regression model over LDA or QDA when assessing patients' samples to quickly determine whether a tumour is benign or malignant. We also found, using Best Subset Selection, that the 7-predictor model provided us with the most accuracy, and we found which 7 variables appeared to be most useful in predictions for classification. This could potentially help doctors and clinicians get a better idea on what to look for and what to place more importance on. For example, by looking at the size of the regression coefficients, doctors can understand that an increase in one of the variables, such as `Mitoses`, is potentially more alarming than an increase in the `Normal.nucleoli`. Importantly, more testing should be done with as many observations as possible to ensure that the conclusions reached here are reproducible - for that purpose, the code throughout this report and the code in the Appendix are available.

7 Bibliography

References

[Hea] Sarah Heaps. Labs on classification methods. Accessible at `https://tmaturi.github.io/ISDS2022/labs_classification.html`
Week 4: Labs.

A R Code for Reproducibility of Experiments

Code is provided by Dr Heaps at [Hea], and slightly altered for our purpose.

```
log_reg_fold_error = function(X, y, test_indices) {

  Xy = data.frame(X, y=y)

  if(ncol(Xy)>1) fold_fit = glm(y ~ ., data=Xy[!test_indices,], family="binomial")
  else fold_fit = glm(y ~ 1, data=Xy[!test_indices,,drop=FALSE], family="binomial")

  p_hat = predict(fold_fit, Xy[test_indices,,drop=FALSE], type="response")
  y_hat = ifelse(p_hat > 0.5, 1, 0)
  y_obs = y[test_indices]
  test_error = 1 - mean(y_obs == y_hat)
  return(test_error)
}

ten_fold_log_cv = function(X,y,fold_indices) {
  #start by building the matrix. X is matrix of predictor variables,
  #Y is vector of response variables.
  Xy = data.frame(X, y)
  #check that every fold has data
  if(!all.equal(sort(unique(fold_indices)), 1:10)) {
    stop("At least one fold is empty.")
  }
  #initialize matrix of fold errors
  fold_error_matrix = matrix(NA, 10, p+1)
  for(fold in 1:10) {
    fold_fit = bestglm(Xy[fold_indices!=fold,], family=binomial, IC="BIC")
    best_models = as.matrix(fold_fit$Subsets[,2:(1+p)])
    for(k in 1:(p+1)) {
      fold_error_matrix[fold, k] = log_reg_fold_error(X[,best_models[k,]], y, fold_indices==fold)
    }
  }
  fold_sizes = numeric(10)
  for(fold in 1:10) {
    fold_sizes[fold] = length(which(fold_indices==fold))
  }
  test_errors = numeric(p+1)
  for(k in 1:(p+1)) {
    test_errors[k] = weighted.mean(fold_error_matrix[,k], w=fold_sizes)
  }
  return(test_errors)
}

ten_fold_cv = function(X, y, fold_indices, fold_error_function) {
  p = ncol(X)
  Xy = cbind(X, y=y)
  if(!all.equal(sort(unique(fold_indices)), 1:10)) stop("At least one fold is empty.")
  fold_errors = numeric(10)
  for(fold in 1:10) {
    fold_errors[fold] = fold_error_function(X, y, fold_indices==fold)
  }
```

```
  fold_sizes = numeric(10)
  for(fold in 1:10) fold_sizes[fold] = length(which(fold_indices==fold))
  test_error = weighted.mean(fold_errors, w=fold_sizes)
  return(test_error)
}


lda_fold_error = function(X, y, test_indices) {
  Xy = data.frame(X, y=y)
  if(ncol(Xy)>1) lda_fit = lda(y ~ ., data=Xy[!test_indices,])
  lda_predict = predict(lda_fit, Xy[test_indices,])
  y_hat = lda_predict$class
  y_obs = y[test_indices]
  test_error = 1 - mean(y_obs == y_hat)
  return(test_error)
}

qda_fold_error = function(X, y, test_indices) {
  Xy = data.frame(X, y=y)
  if(ncol(Xy)>1) qda_fit = qda(y ~ ., data=Xy[!test_indices,])
  qda_predict = predict(qda_fit, Xy[test_indices,])
  y_hat = qda_predict$class
  y_obs = y[test_indices]
  test_error = 1 - mean(y_obs == y_hat)
  return(test_error)
}
```