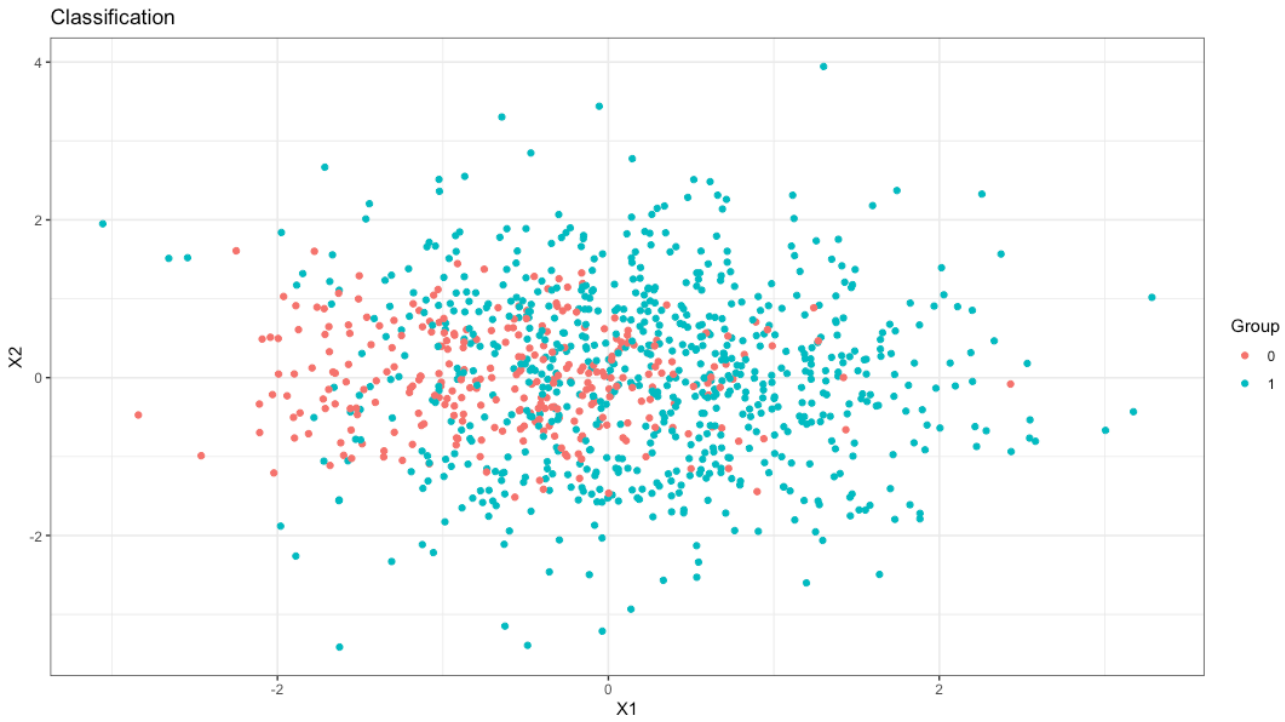
Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070

Assignment 3

Q1)

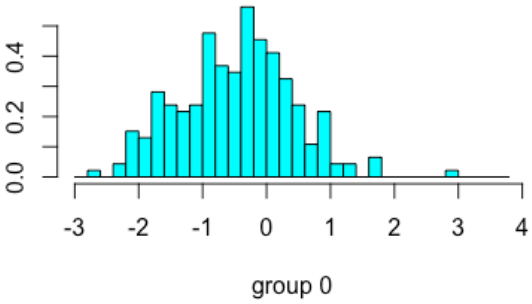


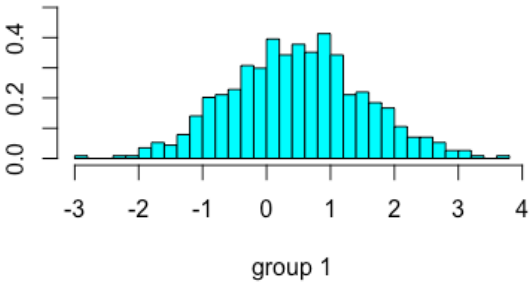
In terms of X1 and X2, there are a thousand of each variable and from the graph it seems that variables in group 1 are more dispersed around the values of X1 and X2, whilst points in group 0 seems to be generally of a smaller range of values. The majority of points are between -2 and 2 for both variables, with only the group 1 points being outside this range. In terms of summary statistics, for X1 the mean is 0.002, with a minimum of -3.056 and a maximum of 3.285. For X2, the mean is 0.016, with a minimum of -3.414 and a maximum of 3.944.

Q2) See code

Q3a) Linear discriminant analysis is a classification technique which aims to find the cutoff at where classification is most effective. This is done by removing the redundant and dependent features by transforming them from higher dimensional space to a space with lower dimensions. So we are essentially trying to maximise the difference between the two groups in our data, by minimising the area of overlap between them. Logistic regression can become unstable when the classes are well separated or when there are few examples from which to estimate the parameters; LDA addresses these issues. LDA assumes that the data is Gaussian and that each attribute has the same variance; predictions are made by estimating the probability that a new set of inputs belongs to each class.

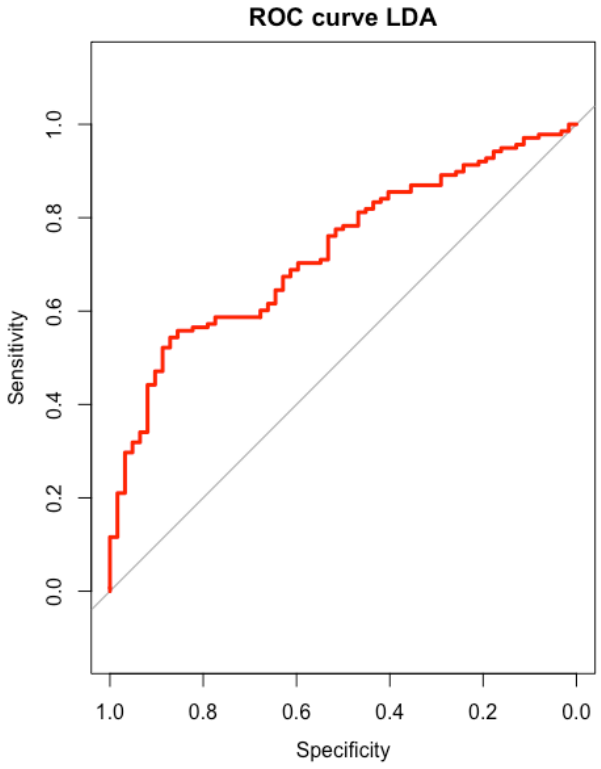
Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070





The following formula was used for LDA: Group~ X1 + X2. The plot on the left shows the distribution of values when LDA was applied to the training set; it seems there is largely a normal distribution of each group. After using LDA to predict on the testing set, a confusion matrix (shown to the right) was generated, which is a measure of a model's classification performance. Of particular interest is the Accuracy, Kappa, sensitivity, specificity, Pos Pred Value, Neg Pred Value and Balanced Accuracy. Accuracy is the fraction of correct classifications, whilst sensitivity is the proportion of true positives predicted as such, and specificity is the equivalent but for negatives. Positive predictive value is the proportion of predicted positives that are actually correct, with negative predictive value being its negative counterpart. Balanced accuracy is likely more useful than accuracy as accuracy is only reliable if the dataset is already balanced, with equal numbers of trials in each class and cross validation fold. Balanced accuracy is simply the mean of the specificity and sensitivity. Kappa is a measure between -1 and 1 of how well the classifier has performed compared to if its performance was random, with 0 or less meaning the classifier is useless. So this model has an accuracy rate of 69%, with balanced accuracy of 58%, neither of which is particularly impressive. Another measure of performance is the receiver operating characteristic (ROC) graph and its area under the curve (AUC). The ROC plots the sensitivity against 1-specificity for a range of probability thresholds; the 45-degree line is a baseline random classifier, so the closer the curve is to the diagonal line, the poorer the classifier model is. The AUC measures how discriminatory the model is, i.e. how well its classifies between classes, with a value of 0.5 indicating no discriminatory power, and a value of 1 indicating perfect discriminatory power, so a higher value is desirable. The ROC curve for this LDA model is shown to the right; the AUC is 0.73. This suggests the model is certainly better than a random

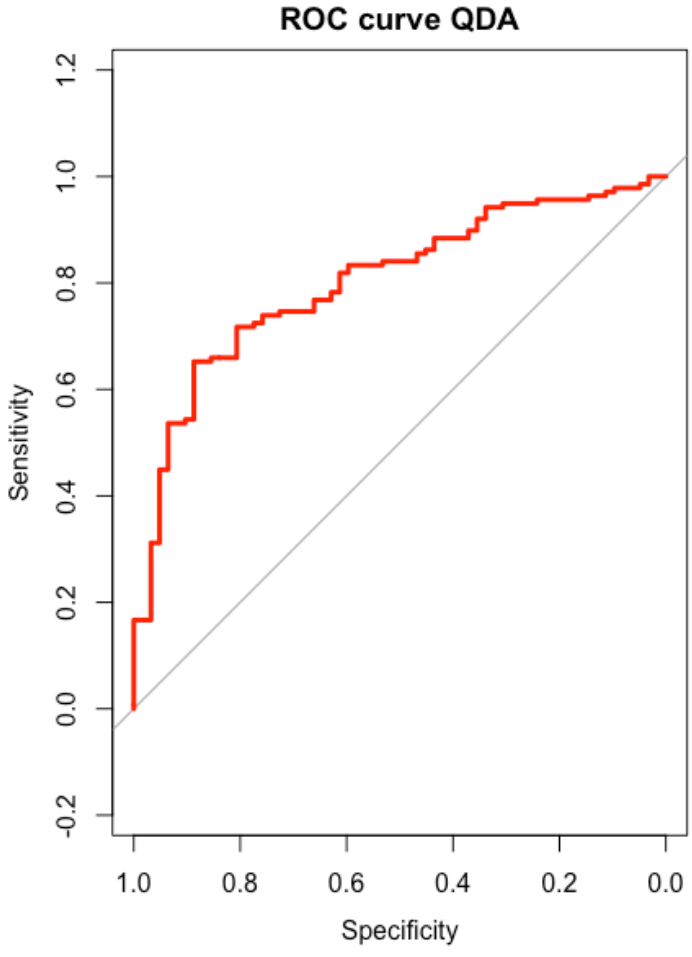
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  18  18
         1  44 120

               Accuracy : 0.69
                 95% CI : (0.6209, 0.7533)
    No Information Rate : 0.69
    P-Value [Acc > NIR] : 0.534290

                  Kappa : 0.1808

 Mcnemar's Test P-Value : 0.001498

            Sensitivity : 0.2903
            Specificity : 0.8696
         Pos Pred Value : 0.5000
         Neg Pred Value : 0.7317
             Prevalence : 0.3100
         Detection Rate : 0.0900
   Detection Prevalence : 0.1800
      Balanced Accuracy : 0.5799

       'Positive' Class : 0
```



system, but not great, whilst the AUC suggests acceptable discriminatory power. It should be noted that the AUC is also helpful as it is scale invariance and classification threshold invariant. This means that it measures how well predictions are ranked rather than absolute values, and also that it measures the quality of the model's predictions regardless of which classification threshold is chosen. Kappa values of 0 or less indicate the classifier is useless; our value of 0.18 suggests the model is not much better than a random system.

b) Quadrant discriminant analysis is an extension of LDA, and is slightly more flexible in that the covariance matrix can be different for each class. QDA is more helpful when the training set is very large, whilst LDA is better for smaller training sets. The confusion matrix for QDA seems to show an improvement compared to LDA in Kappa, accuracy and balanced accuracy (and so by definition sensitivity and specificity). Balanced accuracy of 0.656 suggests that on average the model is classifying better than LDA, and the same is true for accuracy. The Higher Kappa value also suggests the model is classifying better compared to a random system compared to the LDA. This also indicated by the appearance of the ROC curve which is further away from the 45-degree diagonal line than for LDA, and the AUC value is also higher at 0.8, which is very close to an excellent discriminatory power.

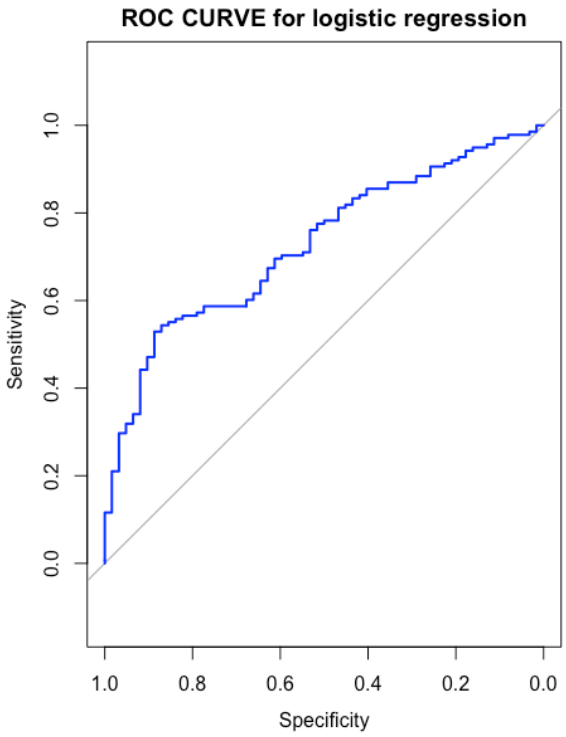
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  27  17
         1  35 121

               Accuracy : 0.74
                 95% CI : (0.6734, 0.7993)
    No Information Rate : 0.69
    P-Value [Acc > NIR] : 0.07152

                  Kappa : 0.3394

 Mcnemar's Test P-Value : 0.01840

            Sensitivity : 0.4355
            Specificity : 0.8768
         Pos Pred Value : 0.6136
         Neg Pred Value : 0.7756
             Prevalence : 0.3100
         Detection Rate : 0.1350
   Detection Prevalence : 0.2200
      Balanced Accuracy : 0.6561

       'Positive' Class : 0
```

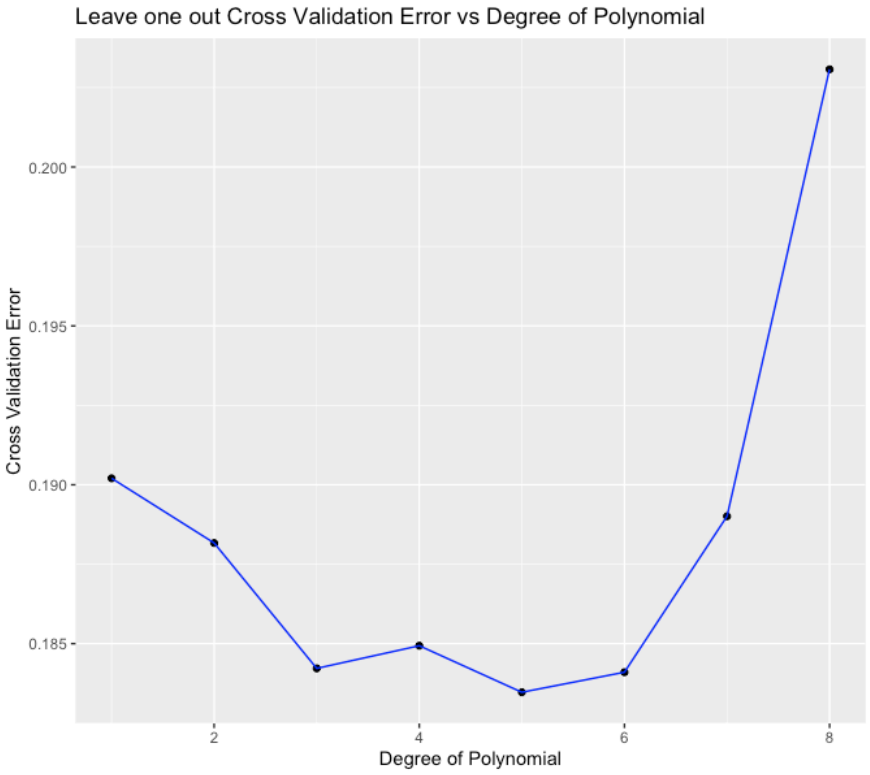


Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070

c) Logistic regression is a statistical machine learning algorithm used for classification and when the response variable is categorical, for example in binomial logistic regression or multinomial logistic regression. It aims to find a relationship between features and probabilities of particular outcomes. It does this by considering the outcome variables on extreme ends of the data and trying to generate a logarithmic line that distinguishes between them. It is a simple, efficient method with low variance, but doesn't handle large numbers of categorical variables well, and also requires the transformation of non linear features. Logistic regressions are also generally not naturally flexible enough to capture more complex relationships. After fitting the logistic regression to the training data and predicting on the test data, the following confusion matrix and ROC curve were produced, with an acceptable AUC of 0.73. As can be seen from the confusion matrix, based on accuracy, balanced accuracy and Kappa, the model is not particularly good at classifying accurately, and is not vastly better than a random system. In the case of logistic regression, in order to optimise our parameters we can use leave one out cross validation, which is a special type of k fold cross validation. K fold cross validation estimates the average validation error that we can expect on new unseen test data by measuring the performance on the part left out in the training process. The average validation error is computed from the K runs, and the hyperparameters that minimise this error are used to build the final model. LOOCV is where N-1 data points are used to train the model, and 1 data point is used to test the model. A plot of the LOOCV error against degree of polynomial is shown to the right; a polynomial of 5 minimises the cross validation error. Running the logistic regression again with a polynomial of 5 results in an improved model as shown by the results below.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  19  18
         1  43 120

               Accuracy : 0.695
                 95% CI : (0.6261, 0.758)
    No Information Rate : 0.69
    P-Value [Acc > NIR] : 0.47339

                  Kappa : 0.198

 Mcnemar's Test P-Value : 0.00212

            Sensitivity : 0.3065
            Specificity : 0.8696
         Pos Pred Value : 0.5135
         Neg Pred Value : 0.7362
             Prevalence : 0.3100
         Detection Rate : 0.0950
   Detection Prevalence : 0.1850
      Balanced Accuracy : 0.5880

       'Positive' Class : 0
```

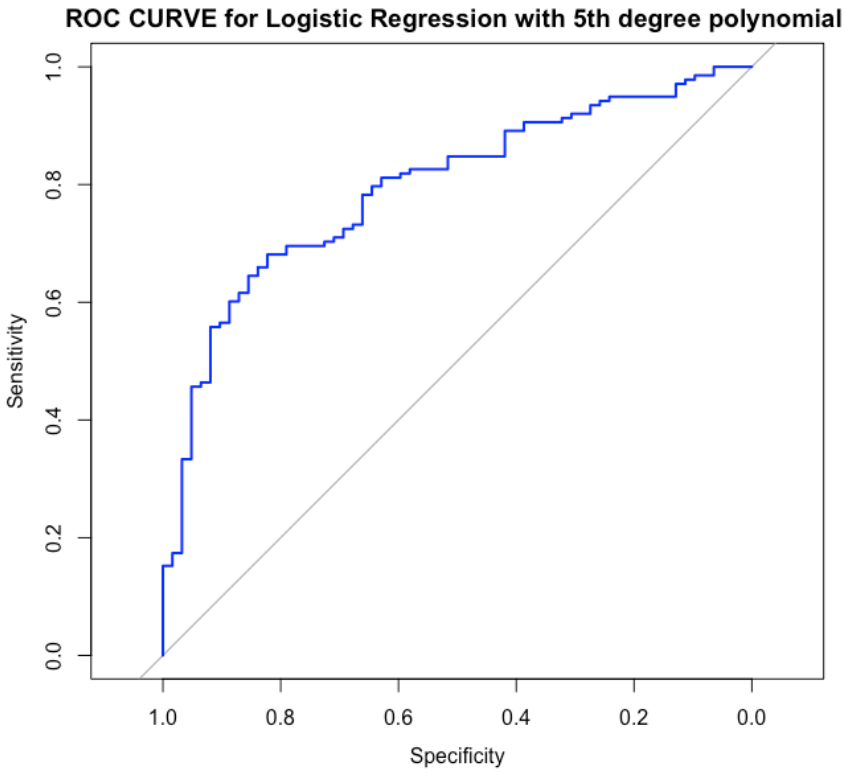




Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070

The balanced accuracy has risen from 0.588 to 0.666, accuracy from 0.695 to 0.735, and Kappa from 0.198 to 0.349. The ROC curve is also further from the 45-degree line suggesting better classification performance has improved, and the AUC has risen from 0.73 to 0.79, indicating improved discriminatory power.

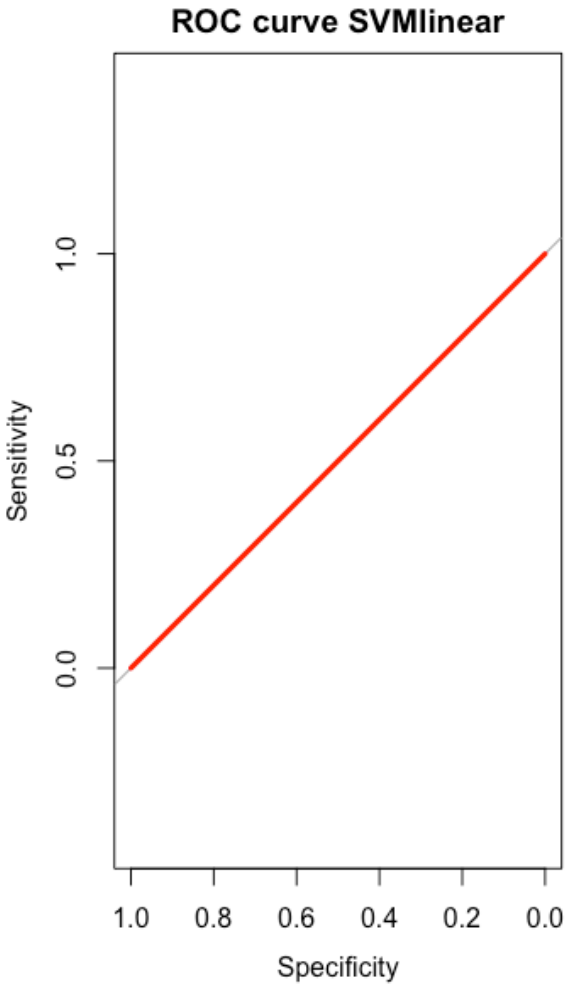
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  30  21
         1  32 117

               Accuracy : 0.735
                 95% CI : (0.6681, 0.7948)
    No Information Rate : 0.69
    P-Value [Acc > NIR] : 0.0956

                  Kappa : 0.3487

 Mcnemar's Test P-Value : 0.1696

            Sensitivity : 0.4839
            Specificity : 0.8478
         Pos Pred Value : 0.5882
         Neg Pred Value : 0.7852
             Prevalence : 0.3100
         Detection Rate : 0.1500
   Detection Prevalence : 0.2550
      Balanced Accuracy : 0.6658

       'Positive' Class : 0
```

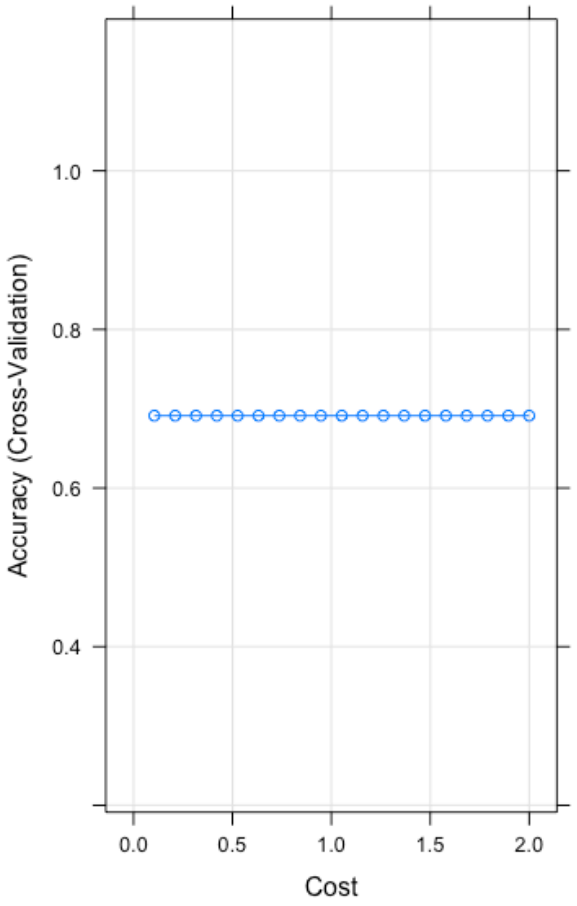


d) Where there is no clear optimal cutoff point where classification is most effective, support vector machines are helpful. Kernels are used to map the original feature set to a higher dimensional space where classes are linearly separable. Usually more dimensions are not what we are looking for, but using kernels allows us to work in an implicit feature space so that the data is not explicitly expressed in higher dimensions. This kernel trick means SVMs are very effective in higher dimensional spaces, even where the number of dimensions is greater than the number of samples. SVMs are also generally very quick and computationally efficient in comparison to to other classification methods, such as k nearest neighbours or neural networks. A kernel function allows us to do a transformation into a new space; to the right and on the following page are the results from fitting an SVM with a linear kernel. As can be seen by the confusion matrix, the results do not appear as they should, with 1 for specificity and 0 for sensitivity; indeed, the Kappa value of 0 indicates the classifier is useless. This is further indicated by the ROC curve on the next page which is directly on the 45-degree diagonal line, so the model is no better than a completely random classifier. As the model is useless, plotting the model accuracy against different values of cost is also not helpful, as the graph shows on the next page.

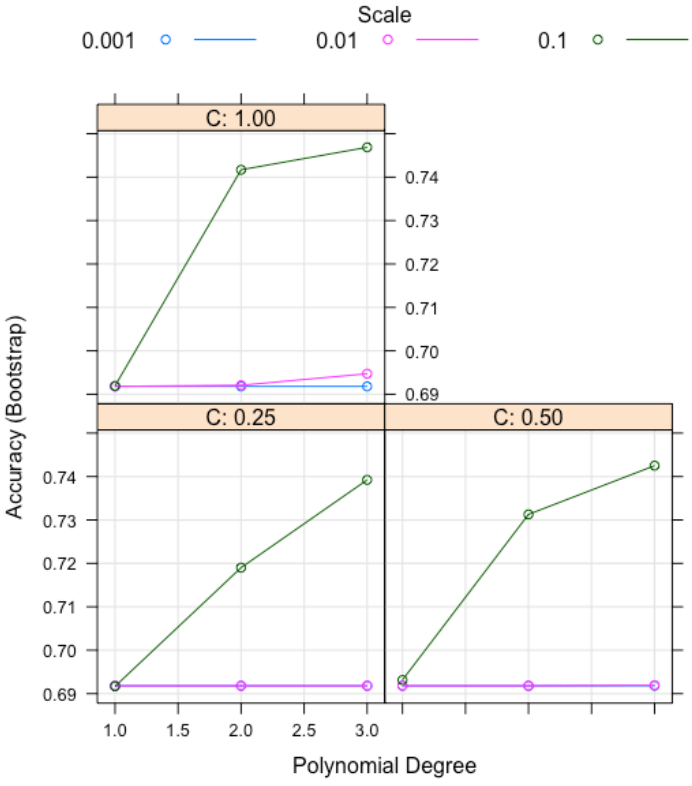
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0   0   0
         1  46 154

               Accuracy : 0.77
                 95% CI : (0.7054, 0.8264)
    No Information Rate : 0.77
    P-Value [Acc > NIR] : 0.5394

                  Kappa : 0

 Mcnemar's Test P-Value : 3.247e-11

            Sensitivity : 0.00
            Specificity : 1.00
         Pos Pred Value :  NaN
         Neg Pred Value : 0.77
             Prevalence : 0.23
         Detection Rate : 0.00
   Detection Prevalence : 0.00
      Balanced Accuracy : 0.50

       'Positive' Class : 0
```

Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070



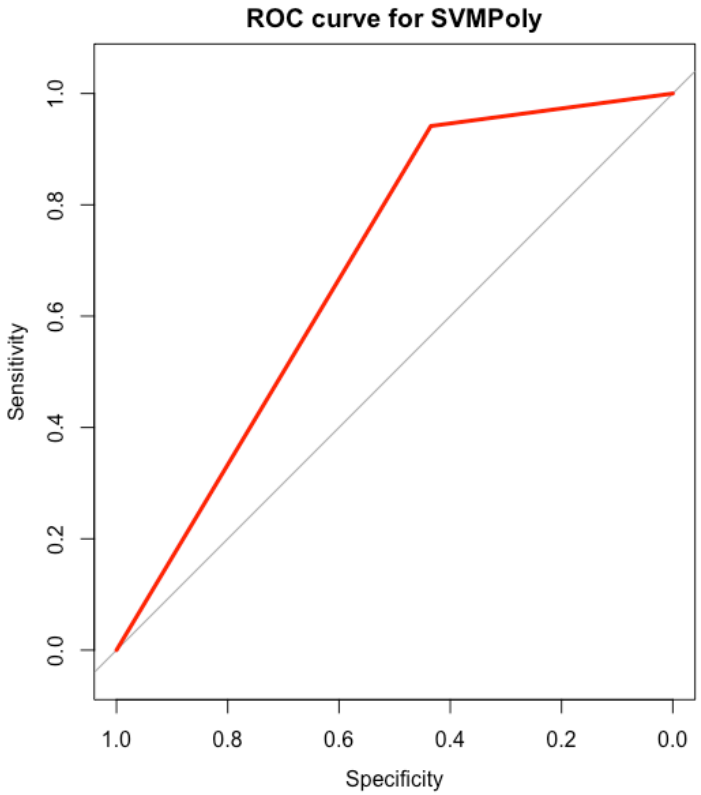


We can adjust the kernel type before fitting the SVM, such as a polynomial (non linear) kernel. The optimal values for the model tuning parameters are automatically chosen, where optimal is defined as values that maximise the model accuracy. The plot to the right shows the model accuracy against different cost values; accuracy was used to select the optimal model using the largest value. The final values used for model optimisation were degree = 3, scale = 0.1 and C = 1, and this is justified by the plot as at these values accuracy is maximised. The top of the next page shows the confusion matrix and ROC curve for the SVMPoly model.



Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070

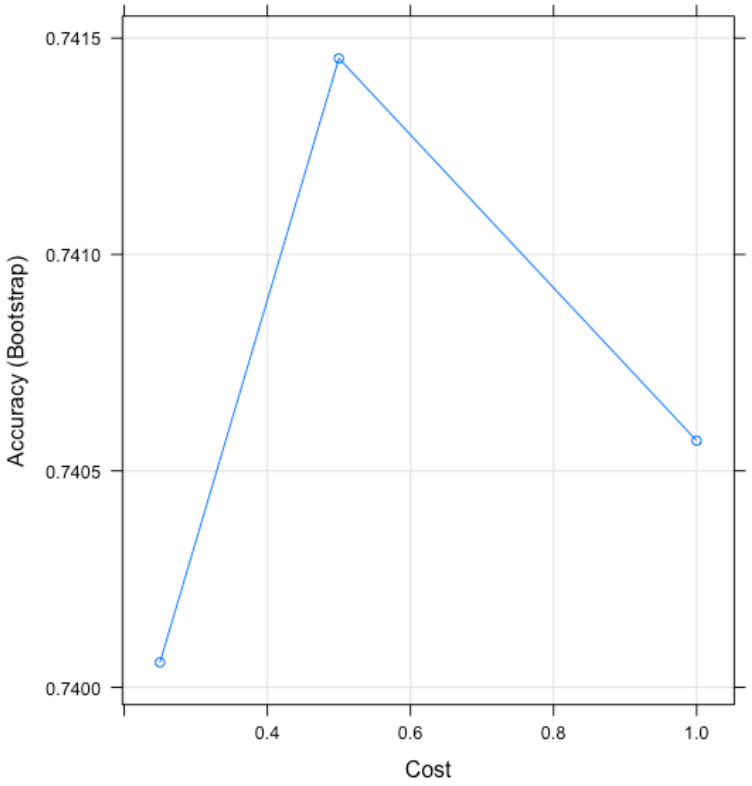
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  20   9
         1  26 145

               Accuracy : 0.825
                 95% CI : (0.7651, 0.875)
    No Information Rate : 0.77
    P-Value [Acc > NIR] : 0.035843

                  Kappa : 0.4324

 Mcnemar's Test P-Value : 0.006841

            Sensitivity : 0.4348
            Specificity : 0.9416
         Pos Pred Value : 0.6897
         Neg Pred Value : 0.8480
             Prevalence : 0.2300
         Detection Rate : 0.1000
   Detection Prevalence : 0.1450
      Balanced Accuracy : 0.6882

       'Positive' Class : 0
```

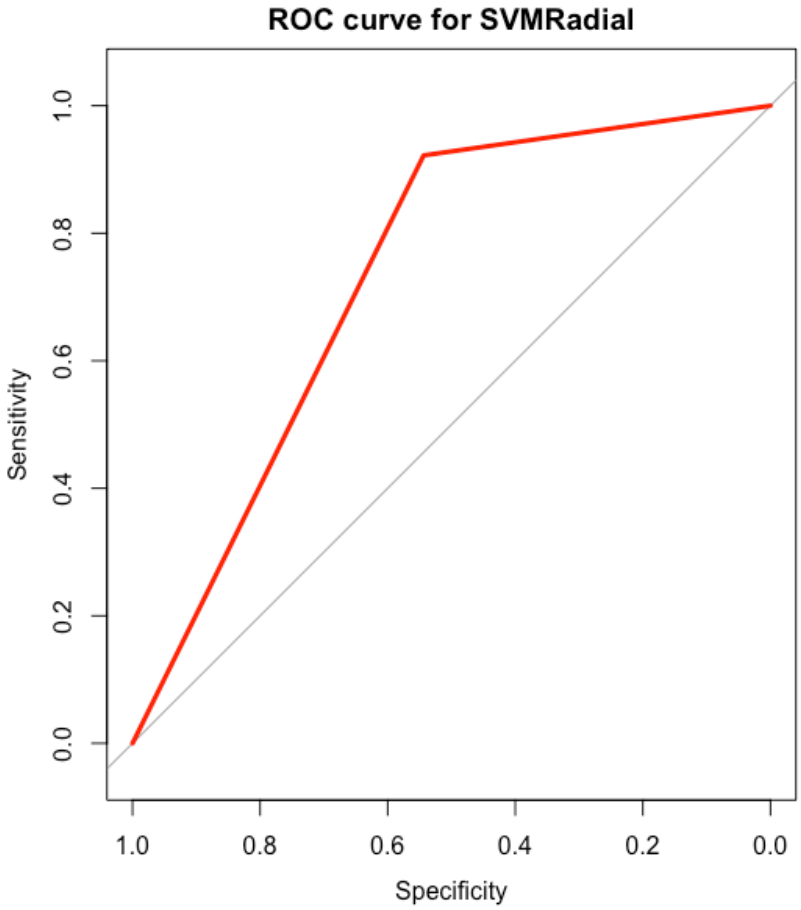


The SVMPoly model works as it should, as the confusion matrix has values that make sense, and no missing Kappa value or strange specificity and sensitivity as the SVMLinear did. The model has a decent accuracy of 0.825, and balanced accuracy of 0.69 (though it could have better sensitivity), and a good Kappa value indicating it is better at classification than a random system by a significant margin. The ROC curve is not on the diagonal line, which is a good sign, though it is interesting to note that the curve is completely smooth compared to those of LDA, QDA and logistic regression. The AUC value though is 0.69, barley outside the acceptable range of discriminatory power; it seems odd that a model with strong accuracy, balanced accuracy and Kappa could have a slightly poor AUC. This may be explained by the very high specificity and low sensitivity, so the model is not particularly good at predicting the proportion of true positives.

Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070

The third type of kernel that can be used is a radial one, the results of which are shown below. In terms of parameter optimisation, tuning parameter 'sigma' was held constant at a value of 1.101 and accuracy was used to select the optimal model using the largest value. The final values used for the model were sigma = 1.101122 and C = 0.5, as indicated by the plot on the right. From the confusion matrix below, the accuracy has risen to 0.835, the balanced accuracy to 0.73, and the Kappa value to nearly 0.5. The specificity has fallen very slightly, but the rise in sensitivity makes up for this. This is a marginal increase in model performance, but an improvement nonetheless; the ROC curve also appears slightly further away from the 45-degree line. Correspondingly, the AUC has risen to 0.73, which indicates acceptable discriminatory power.



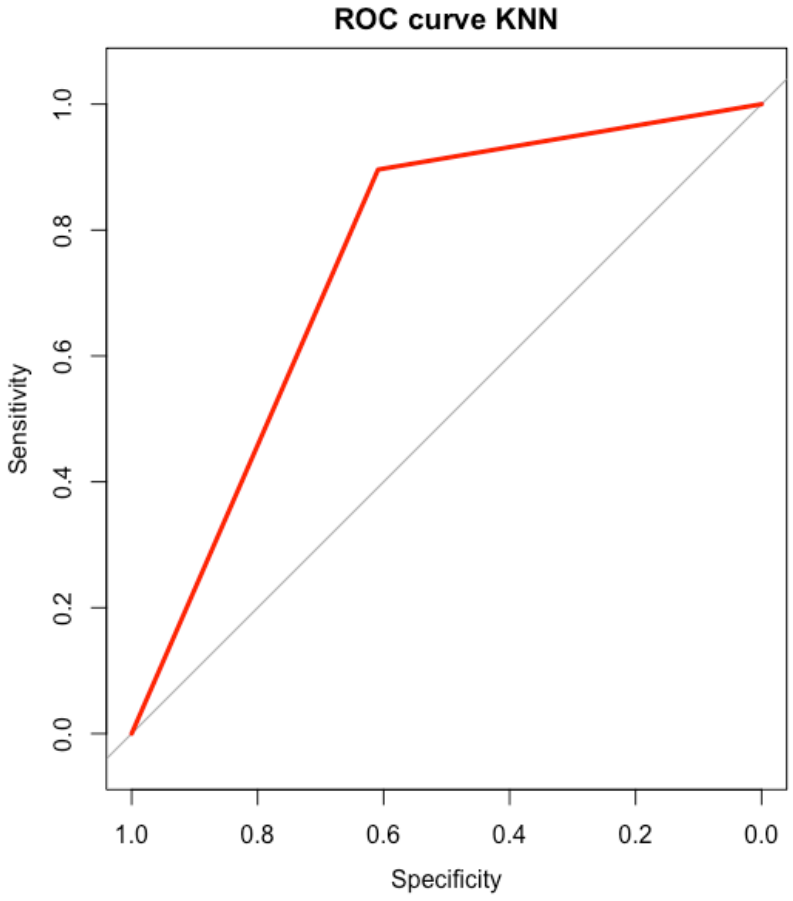



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  25  12
         1  21 142

               Accuracy : 0.835
                 95% CI : (0.7762, 0.8836)
    No Information Rate : 0.77
    P-Value [Acc > NIR] : 0.01534

                  Kappa : 0.4998

 Mcnemar's Test P-Value : 0.16373

            Sensitivity : 0.5435
            Specificity : 0.9221
         Pos Pred Value : 0.6757
         Neg Pred Value : 0.8712
             Prevalence : 0.2300
         Detection Rate : 0.1250
   Detection Prevalence : 0.1850
      Balanced Accuracy : 0.7328

       'Positive' Class : 0
```

Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070

e) The K- nearest neighbours algorithm is one of the simplest models used for classification, and is often used as a baseline to compare various machine learning algorithms. The distance between every training and test data point is computed, before finding the k training points closest to the test point; the test point is then assigned the majority vote of their class label. KNN is useful in that it is a relatively simple algorithm, makes no assumptions about the data, and is versatile as it can be used for both classification and regression. However, it should be noted that it is computationally expensive and is sensitive to irrelevant features and the scale of data. To optimise the parameters, after running KNN on the training data, the accuracy value was used to select the optimal model via 5- fold cross validation which was repeated 10 times; k=11 returned the optimal model. The confusion matrix and ROC curve are shown below. The confusion matrix shows a good accuracy and balanced accuracy of 0.83 and 0.75 respectively, and a strong Kappa value of 0.5, suggesting the model is significantly better than a random system at classifying. The ROC curve is a good distance from the 45-degree line, and the AUC is acceptable 0.75 though one would expect higher discriminatory power as otherwise the model seems very decent.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  28  16
         1  18 138

               Accuracy : 0.83
                 95% CI : (0.7706, 0.8793)
    No Information Rate : 0.77
    P-Value [Acc > NIR] : 0.02383

                  Kappa : 0.5126

 Mcnemar's Test P-Value : 0.86383

            Sensitivity : 0.6087
            Specificity : 0.8961
         Pos Pred Value : 0.6364
         Neg Pred Value : 0.8846
             Prevalence : 0.2300
         Detection Rate : 0.1400
   Detection Prevalence : 0.2200
      Balanced Accuracy : 0.7524

       'Positive' Class : 0
```



4) To pick the best performing model we can compare the ROC curves, AUC values, Kappa values and accuracy/ balanced accuracy values. Looking at the last three measures first, SVMRadial and KNN have the highest accuracy of 0.835 and 0.83 respectively, and balanced accuracy of 0.73 and 0.75 respectively. Kappa and the AUC also differ marginally at 0.499 for SVMRadial and 0.513 for KNN respectively, and 0.73 and 0.75 respectively. The ROC curve for KNN does also appear slightly further away from the 45-degree line than the ROC curve of SVMRadial, suggesting better model performance. The difference in model performance is very slight, but based on these values, as Kappa, balanced accuracy and AUC are higher for KNN, I would pick KNN with the optimal K=11 as the best classification method.

5) After running confusion matrices using the same predictions in question 3 but against the test set of the true data, the accuracy and balanced accuracy of all methods rose, and the AUC also did; the ROC curve also pushed away from the 45-degree line. LDA accuracy rose from 0.69 to 0.8, balanced accuracy from 0.58 to 0.68, and Kappa from 0.18 to 0.4. The AUC also from 0.73 to an excellent 0.85, and the ROC curve was correspondingly further away from the 45-degree line. For QDA the accuracy rose from 0.74 to 0.89, balanced accuracy from 0.656 to 0.835, Kappa from 0.34 to 0.69 and AUC from 0.8 to an outstanding 0.93; the ROC curve also pushed further away from the 45-degree line. Using the same optimised parameters for logistic regression as before, the accuracy rose from 0.74 to 0.87, balanced accuracy from 0.666 to 0.826, Kappa from 0.35 to 0.64 and AUC from 0.79 to an outstanding 0.92; the ROC curve also pushed further away from the 45-degree line. For SVMPoly and SVMRadial, the measurements have improved, but radial is still the superior SVM. SVMradial performance has improved, with accuracy rising from 0.835 to 0.9, the balanced accuracy from 0.73 to 0.83, and the Kappa value from nearly 0.5 to 0.68. The AUC rose from 0.73 to an excellent 0.83, with the ROC curve pushing further away from the 45-degree line. Finally, KNN rose from an accuracy of 0.83 to 0.92, balanced accuracy from 0.75 to 0.9, and Kappa from 0.51 to 0.75. The AUC rose from 0.75 to an outstanding 0.9, and the ROC curve also pushed significantly outwards from the 45-degree line, indicating increased model performance. Taking all these changes into account KNN is still the best performing model, with by far the best accuracy and balanced accuracy, highest Kappa value and joint second highest AUC value.

Daanish Ahsan
Advanced Topics in Statistics
Candidate Number: 124070

Code Appendix

```
library(tidyverse)
library(caret)
library(randomForest)
library(kernlab)
library(MASS)
library(MLeval)
library(pROC)
library(dplyr)
library(verification)
library(boot)
library(reshape2)
library(plotROC)
library(ROCR)

set.seed(123)

#for evaluation

#Regression: root mean squared error (RMSE), R-squared
#Classification: area uder the receiver operating characteristic (ROC)
 curve, confusion matrix

#Actual Questions

Classification <- read.csv('Classification .csv')

#1

ggplot(Classification, aes(x= X1, y= X2)) +
  geom_point(alpha= 1, aes(group= as.factor(Group), colour=
 as.factor(Group))) +
  scale_color_discrete() +
  theme_bw() +
  labs(title= 'Classification', x='X1', y='X2', color='Group')

summary(Classification$X1) #min of -3.056, median of -0.038, mean of 0.002,
 max of 3.285

summary(Classification$X2) #min of -3.414, median of 0.015, mean of 0.016,
 max of 3.944

summary(Classification)

#2

smp_size <- floor(0.8 * nrow(Classification)) #0.8 is for 80% of data to be
 training

train_ind <- sample(seq_len(nrow(Classification)), size = smp_size)
```

```
Ctrain <- Classification[train_ind, ] #creating training set
Ctest <- Classification[-train_ind, ] #creating test set

#5

TrueClassification <- read.csv('ClassificationTrue.csv')

smp_sizetrue <- floor(0.8 * nrow(TrueClassification)) #0.8 is for 80% of
 data to be training

train_indt <- sample(seq_len(nrow(TrueClassification)), size = smp_size)

Ctraintrue <- TrueClassification[train_ind, ] #creating training set
Ctesttrue <- TrueClassification[-train_ind, ]

#compare each methods predictions to the true data in confusion matrix

#3

#need to for each method, describe how it works, results and results of an
 evaluation of methods, and findings

#a), LDA

LDAtrain <- lda(Group~ X1 + X2, data = Ctrain) #LDA on training set

#predictions on test data
testclasspred <- predict(LDAtrain, newdata=Ctest)

plot(LDAtrain)

#evaluation

str(testclasspred)
str(Ctest)
Ctest$Group <- as.factor(Ctest$Group)
testclasspred$class <- as.factor(testclasspred$class)

#Confusion Matrix, predicted vs true for test. We need to compare predicted
 and actual in test data

confusionMatrix(testclasspred$class, Ctest$Group)

#Calculating a confusion matrix can give you an idea of where the
 classification model is right and what types of errors it is making.
#A confusion matrix is used to check the performance of a classification
 model on a set of test data for which the true values are known. Most
 performance measures such as precision, recall are calculated from the
 confusion matrix.
```

```
#Classification accuracy of 73%
#Specificity of 91.55% means 8.5% of values are mislabelled

#q5 confusion matrix w true test data
Ctesttrue$Group <- as.factor(Ctesttrue$Group)

confusionMatrix(testclasspred$class, Ctesttrue$Group) #79.5 accuracy

#ROC w true test data
posteriors <- as.data.frame(testclasspred$posterior)

roc_lda <- roc(response = Ctesttrue$Group, predictor = posteriors['0'])
plot(roc_lda, col="red", lwd=3, main="ROC curve LDA")
auc(roc_lda) #AUC of 90.77

#evaluation- ROC curve on test predictions

#This is probs wrong

roc_lda <- roc(response = Ctest$Group, predictor = posteriors['0'])
plot(roc_lda, col="red", lwd=3, main="ROC curve LDA")
auc(roc_lda)

roc_lda <- roc(response = Ctest$Group, predictor = posteriors['1'])
plot(roc_lda, col="red", lwd=3, main="ROC curve LDA")
auc(roc_lda)

#AUC of 74.8

#Just discuss evaluation of ROC and AUC for test data

#ROC shows:

#the tradeoff between sensitivity and specificity (any increase in
 sensitivity will be accompanied by a decrease in specificity).
#The closer the curve follows the left-hand border and then the top border
 of the ROC space, the more accurate the test.
#The closer the curve comes to the 45-degree diagonal of the ROC space, the
 less accurate the test.

#Sensitivity is true positive rate, specificity is false positive rate

#AUC curve on test data is 0.7354, so this model has acceptable
 discriminatory power?

#AUC ranges in value from 0 to 1.
#A model whose predictions are 100% wrong has an AUC of 0.0;
#one whose predictions are 100% correct has an AUC of 1.0.

#AUC is desirable for the following two reasons:
```

```
#AUC is scale-invariant. It measures how well predictions are ranked,
#rather than their absolute values.

#AUC is classification-threshold-invariant.
#It measures the quality of the model's predictions irrespective of what
 classification threshold is chosen.

#However, both these reasons come with caveats, which may limit the
 usefulness of AUC in certain use cases:

#Scale invariance is not always desirable.
#For example, sometimes we really do need well calibrated probability
 outputs, and AUC won't tell us about that.

#Classification-threshold invariance is not always desirable.
#In cases where there are wide disparities in the cost of false negatives
 vs. false positives, it may be critical to minimize one type of
 classification error.
#For example, when doing email spam detection,
#you likely want to prioritize minimizing false positives (even if that
 results in a significant increase of false negatives).
#AUC isn't a useful metric for this type of optimization.

#b) QDA

QDAtrain <- qda(Group~ X1 + X2, data = Ctrain) #QDA on test set

QDAtestclasspred <- predict(QDAtrain, newdata = Ctest)

#ROC and AUC
QDAposteriors <- as.data.frame(QDAtestclasspred$posterior)

roc_qda <- roc(response = Ctest$Group, predictor = QDAposteriors[,'0'])
plot(roc_qda, col="red", lwd=3, main="ROC curve QDA")
auc(roc_qda) #AUC of 82.5 indicates acceptable discriminatory power from
 model

roc_qda <- roc(response = Ctest$Group, predictor = QDAposteriors[,'1'])
plot(roc_qda, col="red", lwd=3, main="ROC curve QDA")
auc(roc_qda) #same as above

confusionMatrix(QDAtestclasspred$class, Ctest$Group) #accuracy of 79%
#specificity of 88%

#q5 confusion matrix w true test data
confusionMatrix(QDAtestclasspred$class, Ctesttrue$Group) #accuracy of 90%

#roc w true test data

roc_qda <- roc(response = Ctesttrue$Group, predictor = QDAposteriors[,'1'])
plot(roc_qda, col="red", lwd=3, main="ROC curve QDA")
```

```
auc(roc_qda) #AUC of 96.6%

#c) Logistic Regression

#convert ctrain group to factor

Ctrain$Group <- as.factor(Ctrain$Group)
str(Ctrain)
logit <- glm(Group ~ X1+X2,data=Ctrain, family="binomial"(link = 'logit'))

summary(logit)

#stepwise model selection to minimise AIC
logit2 <- stepAIC(logit)
summary(logit2)

summary(logit2$fitted.values)

hist(logit2$fitted.values,main = " Histogram ",xlab = "Values of X1 and
 X2", col = 'light green')

roc(Group~logit2$fitted.values, data = Ctrain, plot = TRUE, main = "ROC
 CURVE", col= "blue")

#prediction on test data only gives numbers, not whole model...?

logitpredict <- predict(logit2, newdata=Ctest, type='response')

hist(logitpredict, main = " Histogram ",xlab = "Values of X1 and X2", col =
 'light green')

roc(Group~logitpredict, data = Ctest, plot = TRUE, main = "ROC CURVE for
 logistic regression", col= "blue") #AUC of 74.8

#do we need a confusion matrix?

n=ifelse(logitpredict>0.5,1,0)
confusionMatrix(as.factor(n),as.factor(Ctest$Group)) #73% accuracy

#do qq plot and that sort of stuff- NEED TO COMMENT

par(mfrow=c(1,2))
fitted.values <- predict(logit2,type="response")
std.deviance.redids <- residuals(logit2,type="deviance")
# Resids vs fitted values
plot(fitted.values,std.deviance.redids,pch=20)
abline(h=0)
# and QQ plot

qqnorm( std.deviance.redids, pch=20 )
qqline( std.deviance.redids )
```

```
#do we need cross validation on this?

set.seed(153)

cv.error=rep(0,10)
# For polynomials 1,2,3... 10 fit a LOOCV model
for (i in 1:10){
  glm.fit=glm(Group ~ poly(X1+X2,i),data=Classification)
  cv.error[i]=cv.glm(Classification,glm.fit)$delta[1]
}
cv.error

folds <- seq(1,10)
df <- data.frame(folds,cvError=cv.error)
ggplot(df,aes(x=folds,y=cvError)) + geom_point() +geom_line(color="blue") +
  xlab("Degree of Polynomial") + ylab("Cross Validation Error") +
  ggtitle("Leave one out Cross Validation - Cross Validation Error vs
 Degree of Polynomial")

#predicting with 5th degree polynomial

polyreg <- glm(Group ~ poly(X1, 5) + poly(X2, 5), data=Ctrain,
 family="binomial"(link = 'logit'))

logitpredictpoly <- predict(polyreg, newdata=Ctest, type='response')

hist(logitpredictpoly, main = " Histogram ",xlab = "Values of X1 and X2",
 col = 'light green')

roc(Group~logitpredictpoly, data = Ctest, plot = TRUE, main = "ROC CURVE
 for Logistic Regression with 5th degree polynomial ", col= "blue") #AUC of
 82.73

#do we need a confusion matrix?

n=ifelse(logitpredictpoly>0.5,1,0)
confusionMatrix(as.factor(n),as.factor(Ctest$Group)) #80% accuracy

#q5 confusion matrix w true test data
confusionMatrix(as.factor(n),as.factor(Ctesttrue$Group)) #90% accuracy

roc(Group~logitpredictpoly, data = Ctesttrue, plot = TRUE, main = "ROC
 CURVE for Logistic Regression with true test data", col= "blue") #AUC of
 96.7

#explain it

#d SVM

set.seed(103) # for reproducibility
```

```
ii <- createDataPartition(Classification[, 3], p=.8, list=F) ## returns
 indices for train data
xTrain <- Classification[ii, 1:2]; yTrain <- Classification[ii, 3]
xTest <- Classification[-ii, 1:2]; yTest <- Classification[-ii, 3]

yTrain <- as.factor(yTrain)
yTest <- as.factor(yTest)

mdl <- train(x=xTrain,y=yTrain, method='svmLinear')
print(mdl)
plot(mdl)
# Test model on testing data
yTestPred <- predict(mdl, newdata=xTest)
confusionMatrix(yTestPred, yTest) # does not seem correct

yTestPred <- as.numeric(yTestPred)

roc_qda <- roc(response = yTest, predictor = yTestPred)
plot(roc_qda, col="red", lwd=3, main="ROC curve for SVMLinear")
auc(roc_qda)

#By default the SVM linear classifier is built using C = 1. It's possible
 to automatically compute SVM for
#different values of C and to choose the optimal one that maximise the
 model cross-validation accuracy. In
#the following example we use a 5-fold cross validation, and try 20
 different C values between 0 and 2.

mdl <- train(x=xTrain,y=yTrain, method = "svmLinear",
             trControl = trainControl("cv", number = 5),
             tuneGrid = expand.grid(C = seq(0, 2, length = 20)))

print(mdl)

# Plot model accuracy vs different values of Cost
plot(mdl)
# Print the best tuning parameter C that maximises model accuracy
mdl$bestTune
# Test model on testing data
yTestPred <- predict(mdl, newdata=xTest)
confusionMatrix(yTestPred, yTest) # predicted/true, #its just the same as
 before, doesnt seem right

yTestPred <- as.numeric(yTestPred)

roc_qda <- roc(response = yTest, predictor = yTestPred)
plot(roc_qda, col="red", lwd=3, main="ROC curve SVMlinear")
auc(roc_qda)

#Next we fit a SVM using a non-linear (polynomial) kernel. The package
 automatically chooses the optimal
```

```
#values for the model tuning parameters, where optimal is defined as values
 that maximise the model accuracy.

mdl <- train(x=xTrain,y=yTrain, method='svmPoly')
print(mdl)
plot(mdl)

# Test model on testing data
yTestPred <- predict(mdl, newdata=xTest)

confusionMatrix(yTestPred, yTest) # predicted/true #accuracy of 82.5%, this
 seems better

yTestPred <- as.numeric(yTestPred) #dont run this if we want confusion
 matrix in below bit for true test data

roc_svmpoly <- roc(response = yTest, predictor = yTestPred)
plot(roc_svmpoly, col="red", lwd=3, main="ROC curve for SVMPoly")
auc(roc_svmpoly) #68% AUC

#svm poly but with true test data

set.seed(103) # for reproducibility
iit <- createDataPartition(TrueClassification[, 3], p=.8, list=F) ##
 returns indices for train data
xTraint <- TrueClassification[ii, 1:2]; yTraint <- TrueClassification[ii,
 3]
xTestt <- TrueClassification[-ii, 1:2]; yTestt <- TrueClassification[-ii,
 3]

yTraint <- as.factor(yTraint)
yTestt <- as.factor(yTestt)

confusionMatrix(yTestPred, yTestt) # predicted/true #accuracy of 89.5

#Roc w real data

yTestPred <- as.numeric(yTestPred)

roc_svmpoly <- roc(response = yTestt, predictor = yTestPred)
plot(roc_svmpoly, col="red", lwd=3, main="ROC curve SVMPoly with true test
 data")
auc(roc_svmpoly) #AUC stays same

#repeating but with svmradial

mdl <- train(x=xTrain,y=yTrain, method='svmRadial')
print(mdl)
plot(mdl)

# Test model on testing data
```

```r
yTestPred <- predict(mdl, newdata=xTest)

confusionMatrix(yTestPred, yTest) # predicted/true #accuracy of 84%

yTestPred <- as.numeric(yTestPred)

roc_svmradial <- roc(response = yTest, predictor = yTestPred)
plot(roc_svmradial, col="red", lwd=3, main="ROC curve for SVMRadial")
auc(roc_svmradial) #AUC of 0.7339

#svm radial but w true test data

confusionMatrix(yTestPred, yTestt) # predicted/true #accuracy of 89.5%

yTestPred <- as.numeric(yTestPred)

roc_svmradial <- roc(response = yTestt, predictor = yTestPred)
plot(roc_svmradial, col="red", lwd=3, main="ROC curve for SVMRadial")
auc(roc_svmradial) #AUC remains same

#e K nearest neighbours

set.seed(103) # for reproducibility
ii <- createDataPartition(Classification[, 3], p=.8, list=F) ## returns
 indices for train data
xTrain <- Classification[ii, 1:2]; yTrain <- Classification[ii, 3]
xTest <- Classification[-ii, 1:2]; yTest <- Classification[-ii, 3]
dim(xTrain)
dim(xTest)

yTrain <- as.factor(yTrain)
yTest <- as.factor(yTest)

opts <- trainControl(method='repeatedcv', number=5, repeats=10, p=0.8)

set.seed(1040) # for reproducibility
mdl <- train(x=xTrain, y=yTrain, # training data
             method='knn', # machine learning model
             trControl=opts, # training options
             tuneGrid=data.frame(k=seq(2, 15))) # range of k's to try

print(mdl) #optimal k is 11

yTestPred <- predict(mdl, newdata=xTest)
confusionMatrix(yTestPred, yTest) #80% accuracy

yTestPred <- as.numeric(yTestPred)

roc_knn <- roc(response = yTest, predictor = yTestPred)
plot(roc_knn, col="red", lwd=3, main="ROC curve KNN")
auc(roc_knn) #AUC of 0.7524
```

```
#Repeating but for real data

confusionMatrix(yTestPred, yTestt) #89.5% accuracy

yTestPred <- as.numeric(yTestPred)

roc_knn <- roc(response = yTestt, predictor = yTestPred)
plot(roc_knn, col="red", lwd=3, main="ROC curve LDA")
auc(roc_knn) #AUC remains same
```