Classification Using Multidimensional Cancer Data
STAT 388/488 Final Project: Spring 2020

1 Abstract

One of the most common goals in statistical machine learning is classification. In this report I assess two classification algorithms, quadratic discriminant analysis and random forests, to classify tumors. Here I demonstrate that in a breast cancer data set with more than 30 variables, both of these techniques can reliably classify tumors as benign or malignant. I also perform quadratic discriminant analysis on the first two principal components without sacrificing too much accuracy.

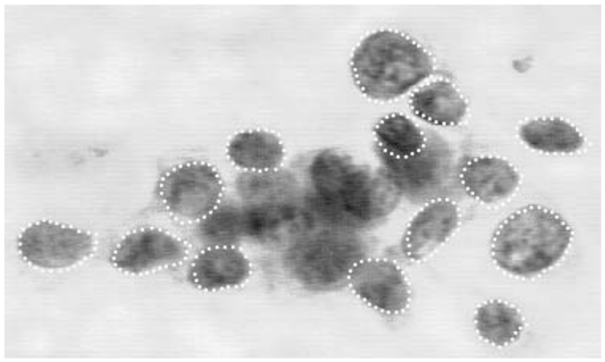
2 Summary of the Data

The Wisconsion diagnostic breast cancer (WDBC) data set was developed in 1993 at the University of Wisconsin–Madison. The data set was a collaboration between the General Surgery Department, which provided images, and the Department of Computer Sciences, which performed feature extraction on the images.

First, researchers recorded 569 images of benign and malignant tumor cells by optaining fine needle aspirates from breast cancer tumors. Then, they used computer vision to label the boundaries of cell nuclei within each image. These boundaries were used for feature extraction. The goal was to identify diagnostic criteria to classify tumors as benign or malignant.

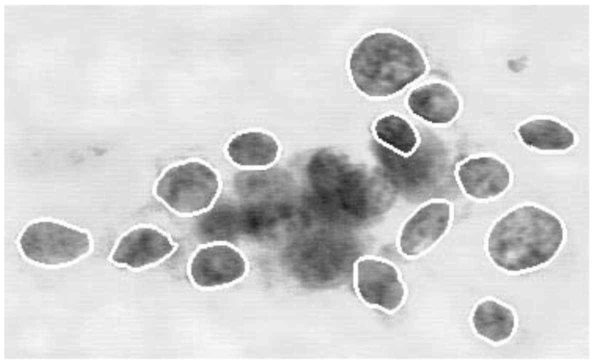
Figure 1 shows a sample image of cells from a breast cancer patient. **Figure 2** shows the same image after fitting contours to each nucleus with the aid of computer vision software.

Figure 1



Sample image used to prepare the WDBC data set. The boundaries of cell nuclei are fitted with splines.

Figure 2



Deformable splines converge to form contours. These contours are the basis for feature extraction.

From each image (n = 569), researchers measured 10 major features. The technical details are given in the appendix (see **Supplementary Table 1**). Researchers computed the mean, standard error, and *mean of the cells with the three highest values* for each feature, resulting in a diagnostic panel of 30 features per image.

To simplify data analysis, I removed the index from the data set, which did not affect predictions.

After importing data and removing the index, V1 encoded the class (1 = benign / 2 = malignant). The class distribution was fairly balanced, with 357 benign and 212 malignant tumors. Each of the 30 features was a real-valued continuous variable. In R, the first 6 observations had the following values:

```
> head(m)
```

```
     V1    V2    V3     V4     V5      V6      V7     V8      V9    V10     V11    V12    V13   V14
[1,]  2 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710 0.2419 0.07871 1.0950 0.9053 8.589
[2,]  2 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017 0.1812 0.05667 0.5435 0.7339 3.398
[3,]  2 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790 0.2069 0.05999 0.7456 0.7869 4.585
[4,]  2 11.42 20.38  77.58  386.1 0.14250 0.28390 0.2414 0.10520 0.2597 0.09744 0.4956 1.1560 3.445
[5,]  2 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430 0.1809 0.05883 0.7572 0.7813 5.438
[6,]  2 12.45 15.70  82.57  477.1 0.12780 0.17000 0.1578 0.08089 0.2087 0.07613 0.3345 0.8902 2.217

     V15      V16     V17     V18     V19     V20      V21   V22   V23    V24    V25    V26    V27
[1,] 153.40 0.006399 0.04904 0.05373 0.01587 0.03003 0.006193 25.38 17.33 184.60 2019.0 0.1622 0.6656
[2,]  74.08 0.005225 0.01308 0.01860 0.01340 0.01389 0.003532 24.99 23.41 158.80 1956.0 0.1238 0.1866
[3,]  94.03 0.006150 0.04006 0.03832 0.02058 0.02250 0.004571 23.57 25.53 152.50 1709.0 0.1444 0.4245
[4,]  27.23 0.009110 0.07458 0.05661 0.01867 0.05963 0.009208 14.91 26.50  98.87  567.7 0.2098 0.8663
[5,]  94.44 0.011490 0.02461 0.05688 0.01885 0.01756 0.005115 22.54 16.67 152.20 1575.0 0.1374 0.2050
[6,]  27.19 0.007510 0.03345 0.03672 0.01137 0.02165 0.005082 15.47 23.75 103.40  741.6 0.1791 0.5249

     V28    V29    V30     V31
[1,] 0.7119 0.2654 0.4601 0.11890
[2,] 0.2416 0.1860 0.2750 0.08902
[3,] 0.4504 0.2430 0.3613 0.08758
[4,] 0.6869 0.2575 0.6638 0.17300
[5,] 0.4000 0.1625 0.2364 0.07678
[6,] 0.5355 0.1741 0.3985 0.12440
```

I investigated the structure of the data by preparing a boxplot of the features (**Figure 3**) and a boxplot of the centered and scaled features (**Figure 4**).

<u>Figure 3</u>

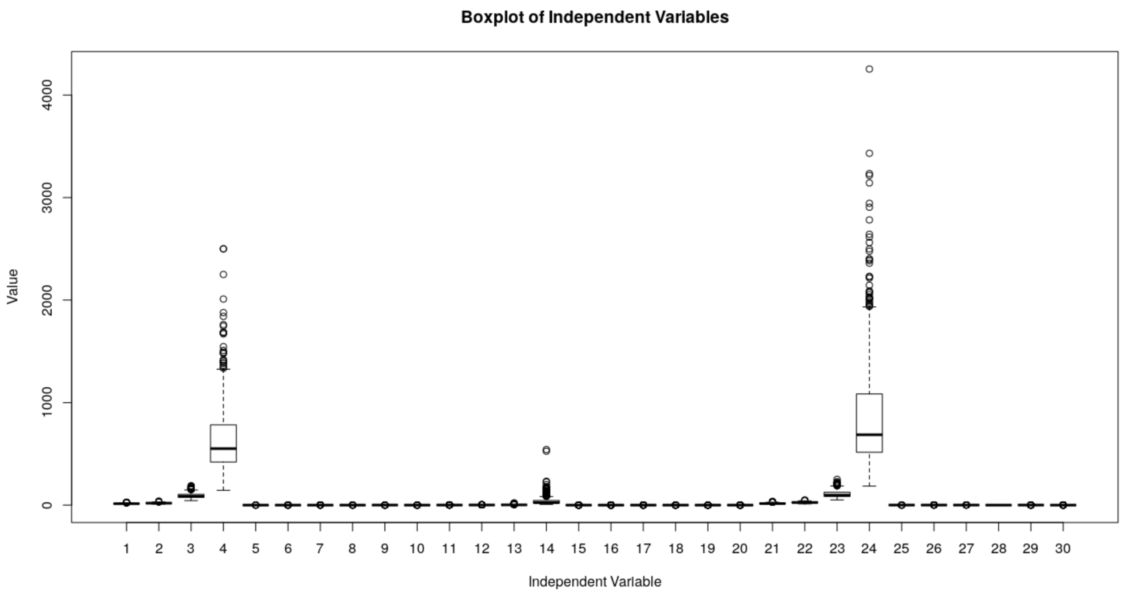
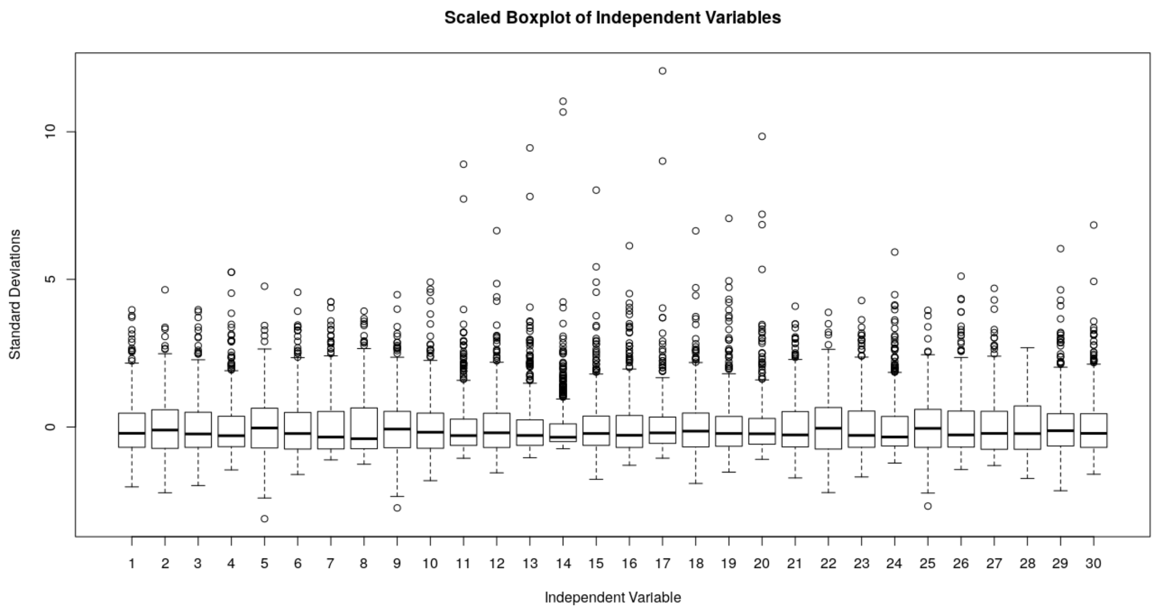
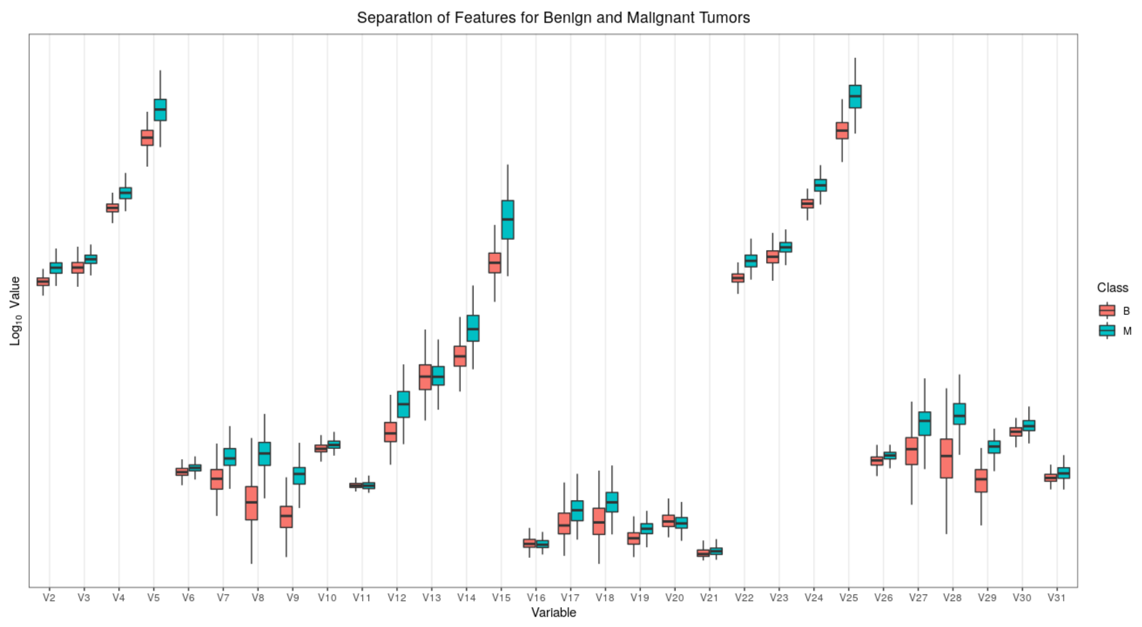


Figure 4



To assess differences between tumor class, I prepared a grouped boxplot (**Figure 5**) and logged variables to examine their differences on a smaller scale. On average, malignant tumors (M) had higher values of each feature than benign tumors (B).

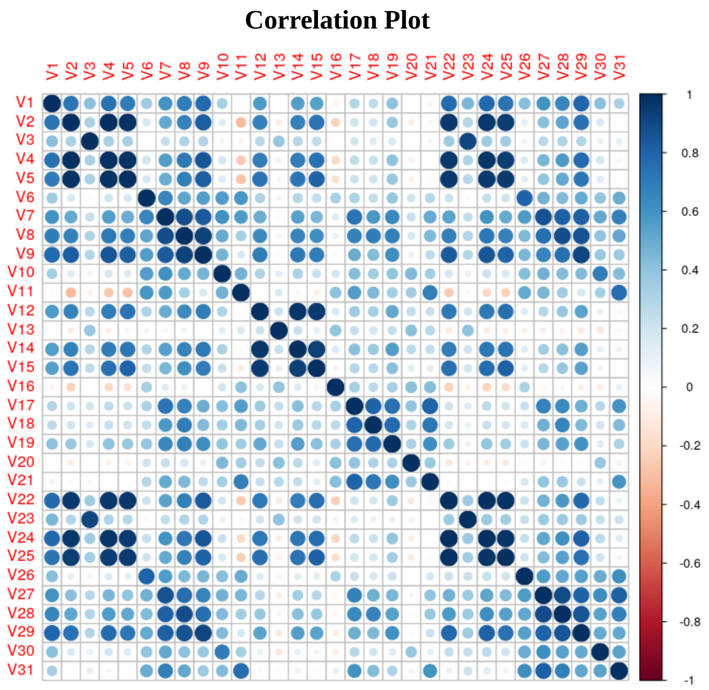
Figure 5



I noted that the apparent difference in feature distribution could be useful for classification.

To investigate correlations in the data, I also produced a correlation plot (**Figure 6**), visualizing the direction and magnitude of correlations between variables. I was especially interested in column 1, which encoded tumor class. I observed that many variables were correlated with one another, which is logical, because some variables are functions of others.

Figure 6



I suspected these correlations would be useful for classification.

3 Analysis and Results

After seeing the data, my objective was to create one or more classification models to classify tumors as benign or malignant using the available data. I did not have to do any data cleaning.

As a first step, I tested for a difference in mean vectors. This decision was motivated by grouped boxplot, which showed differences between tumor classes. At a confidence level of 0.95, Hotelling's 2-sample T^2 test was significant, $T^2 = 61.53$, $p < 2.2 \times 10^{-16}$. Thus, according to the test, the mean vectors of image features differed between benign and malignant tumors. Since I did not investigate the assumptions of multivariate normality or homogeneity of variance (yet), these results were tentative, but they provided some evidence that the features might help to establish classification rules.

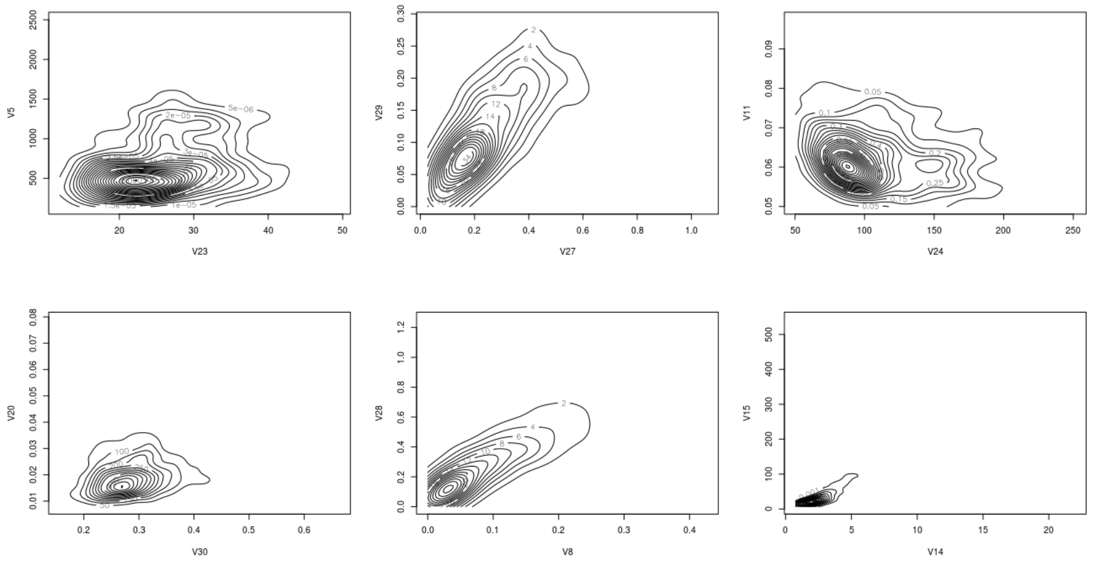
The first classification model I considered was discriminant analysis. While linear discriminant analysis (LDA) assumes equal covariances between groups, quadratic discriminant analysis (QDA) does not. Since the variance-covariance matrices differed between classes, I decided to use QDA.

First, I assessed the assumption of multivariate normality. Like many inferential methods in multivariate analysis, QDA assumes variables are independent and multivariate normal (MVN) in distribution. I used

Royston's test to evaluate the MVN assumption. I also evaluated the MVN assumption graphically by visualizing a sample of contour plots (**Figure 7**).

Figure 7

Bivariate Contour Plots of a Random Subset of Variables



Royston's test determined that data were not MVN, and the bivariate contour plots supported this assessment. Several variables were highly skewed.

In addition, I calculated variance inflation factor (VIF) to investigate multicollinearity, which is sometimes problematic in discriminant analysis (Næs & Mevik, 2001). I called the `vif` function on a simple logit model in R, and found that many variables were highly collinear, VIF greater than 1000 in many cases.

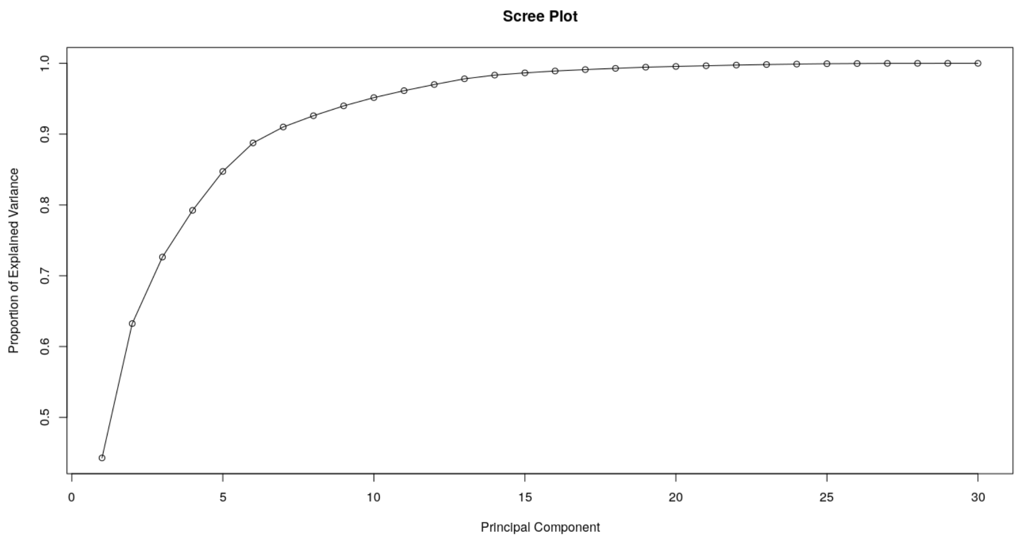
While the data appeared to violate certain assumptions, I attempted QDA anyway. It has been reported that QDA is relatively robust to violations of normality, except for large violations (Lantz, 2019).

As an exercise in dimension reduction, I also performed PCA. PCA can remedy violations of assumptions by defining new variables. However, mainly I wanted to see if a small number of components could explain a high proportion of variance.

After PCA, the proportion of variance in the data explained by the first two principal components was 63.2%. **Figure 8** shows a scree plot. The first two principal components alone explained more than half the variability in the data, making PCA a powerful tool for dimension reduction for this data set.

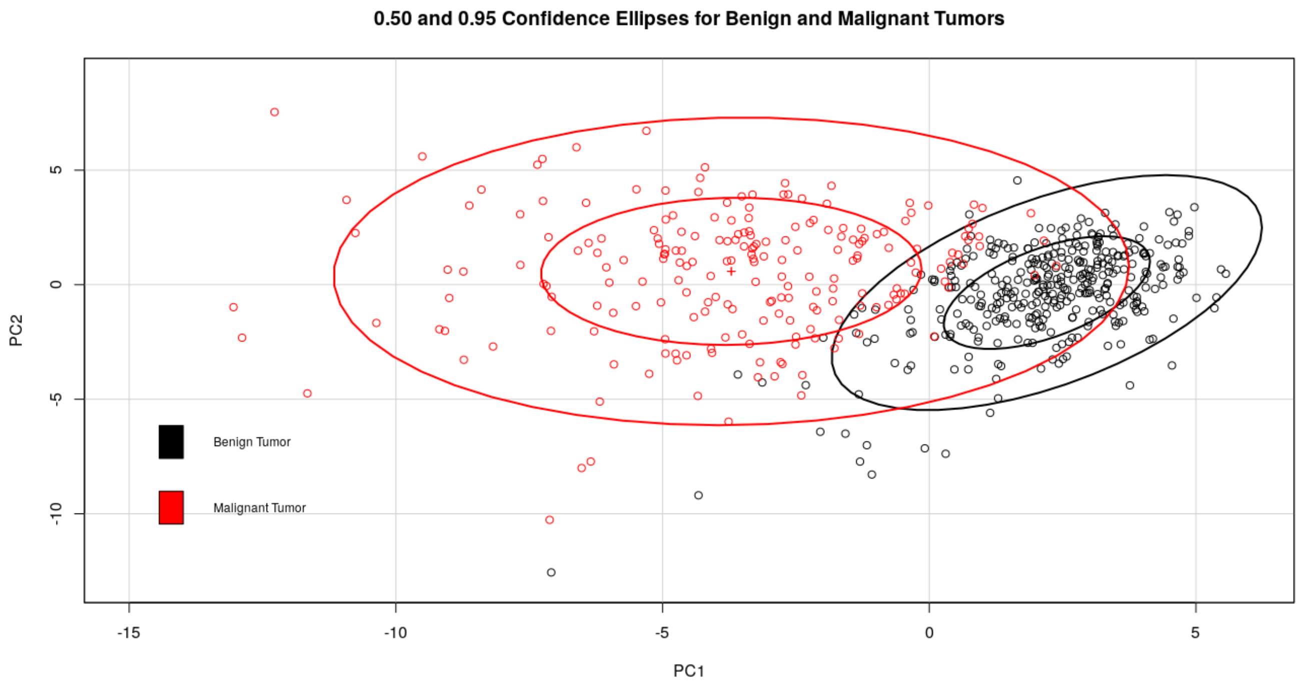
I also plotted the projections of the data onto PC1 and PC2 (**Figure 9**), and the two tumor classes seemed to separate.

Figure 8



The cumulative proportion of variance explained by the first two principal components was 63.2%.

Figure 9



Because PC1 and PC2 encoded class differences, I hypothesized that QDA could perform well in classification given the first two principal components as inputs. Since classes were mostly balanced, I let the prior probabilities be 0.5 for each class.

After training QDA on PC1 and PC2, the model had an apparent rate (APER) of 5.62%. The confusion matrix is shown below:

```
> CM <- table(m[,1], predict(qda.pca, pc.dat)$class); CM # confusion matrix, apparent error rate

      1   2
  1 339  18
  2  14 198
```

Using the holdout (Lachenbruch's) method, I calculated the expectation of actual error rate (E(AER)) as 5.98%, only marginally higher. The confusion matrix is shown below:

```
> CM <- table(m[,1], holdout.class); CM # confusion matrix, expectation of actual error rate

      1   2
  1 337  20
  2  14 198
```

These results were promising: they demonstrated that QDA could succeed in classification using only the first 2 principal components.

Next, I investigated whether a QDA model trained using all the available data would perform even better. Indeed, this model was even more accurate with an APER of 2.46% and E(AER) of 4.39%, confusion matrices printed below.

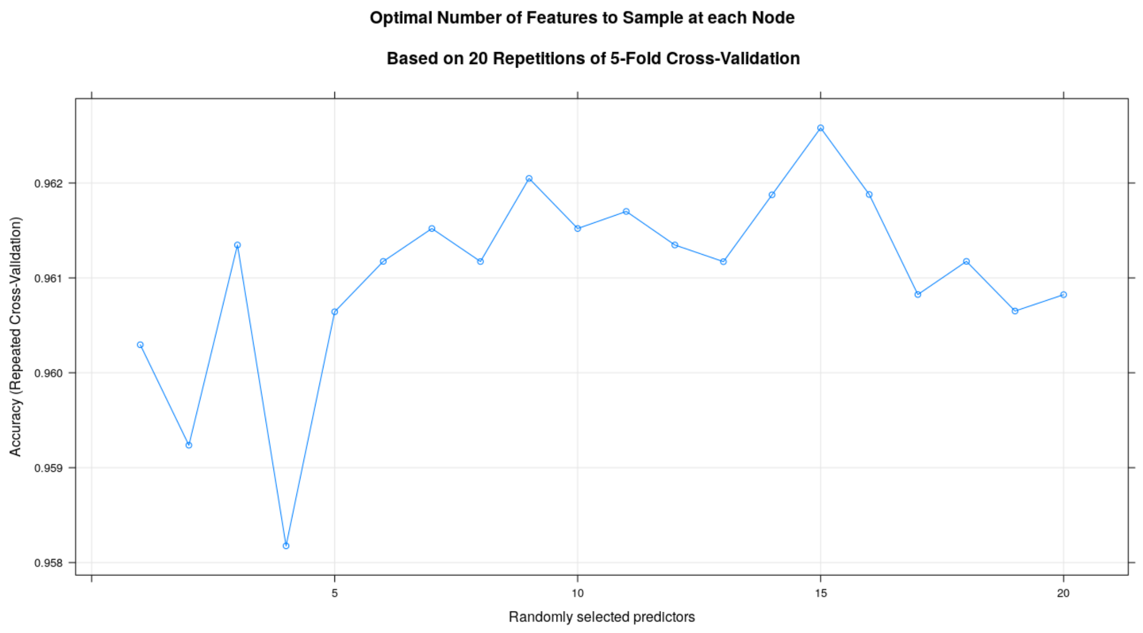
```
> CM <- table(m[,1], predict(qda, wdbc)$class); CM # confusion matrix, apparent error rate

      1   2
  1 352   5
  2   9 203

> CM <- table(m[,1], holdout.class); CM # confusion matrix, expectation of actual error rate

   holdout.class
      1   2
  1 345  12
  2  13 199
```

Finally, in order to develop a model robust to violations of MVN and multicollinearity, I trained random forests. Random forests are nonparametric and thus relatively insensitive to skweness and kurtosis or the presence of collinear features.

One benefit of the random forest algortihm is that error estimates can be obtained without jackknifing or cross-validation. Since trees are grown on bootstrapped samples, observations from the out-of-bag samples can be used to assess the quality of predictions. The misclassification rate obtained in this manner is called out-of-bag (OOB) error.

Random forests often perform well with default parameters, but authors like Li (2013) have found that parameter tuning can improve accuracy. I called the `caret` library and `randomForest` algorithm to tune parameters, varying the number of features to sample at each node from 1 to 20. For each iteration, I averaged the OOB error across 20 repetitions of 5-fold cross validation. Results are plotted in **Figure 10**.

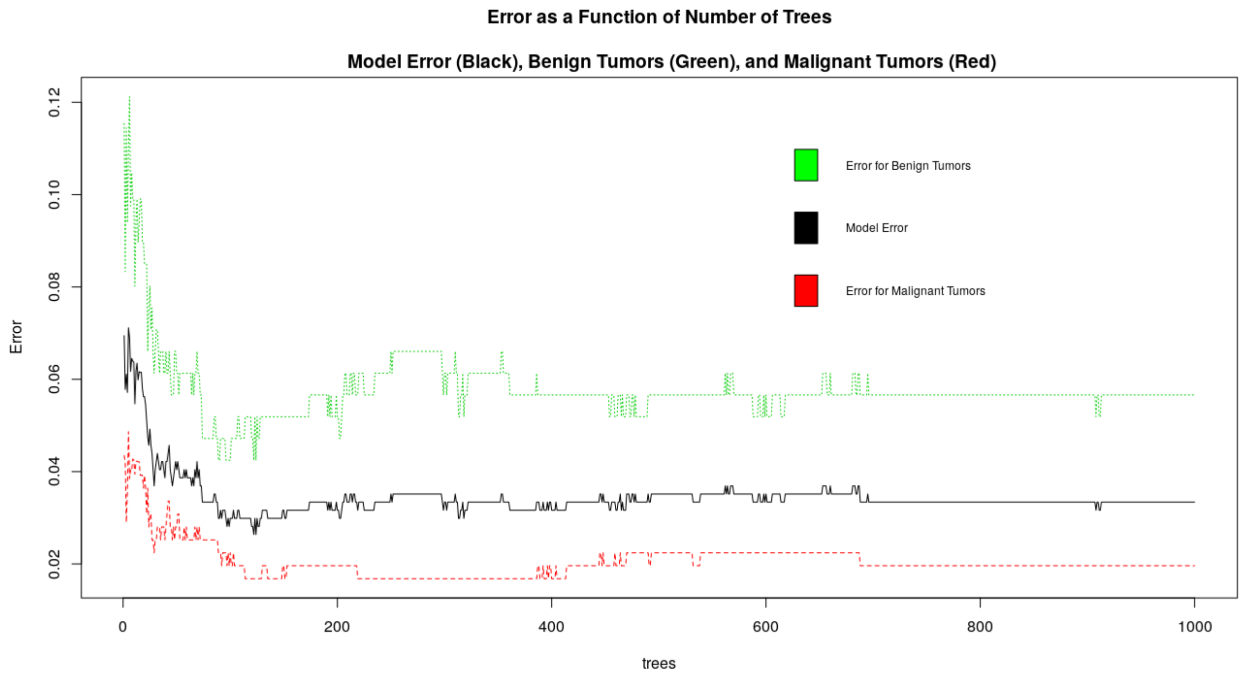
Figure 10



The parameter search determined that the optimal number of features to sample at each node was 15, as this was the value that minimized OOB error.

I also investigated a reasonable number of trees to grow. Results are plotted in **Figure 11**.

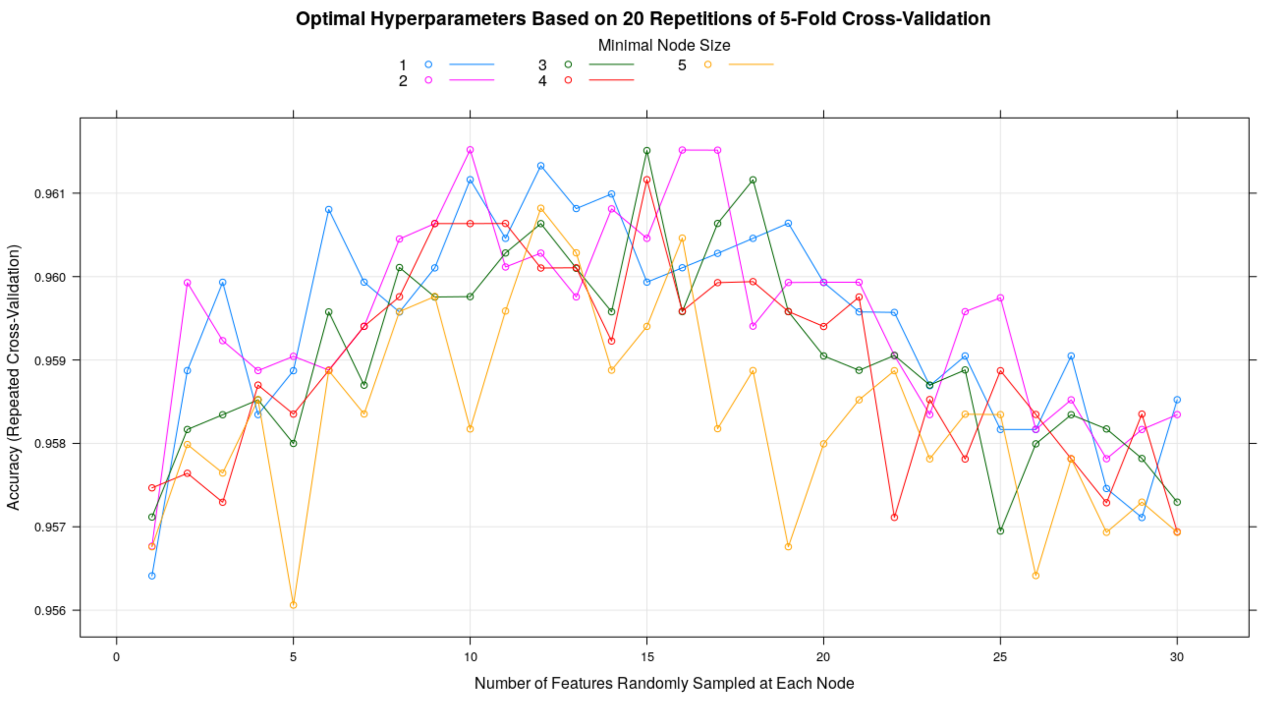
Figure 11



It appeared the OOB prediction error converged (due to the strong law of large numbers) as the number of trees reached 300 or so. I chose 400 as a reasonable number of trees to grow.

To test parameters using another algorithm, I also called the `ranger` package, a fast implementation of `randomForest`, especially for high-dimensional data (Wright & Ziegler, 2017). This time, I tested the optimal number of features to sample at each node as well as minimum node size. The results are plotted in **Figure 12**.

Figure 12



I found that the ideal minimum node size was 2 and the optimal number of features to sample at each node was 10, which differed from the previous finding. The discrepancy in optimal number of features sampled at each node was not concerning, because the `ranger` package implements a slightly different algorithm, and random forest is a stochastic process. Altogether, it seemed that a range of parameters was reasonable.

Using 400 trees in each ensemble, I declared random forest models using all available predictors, 30 in total. I tested both the `randomForest` function and `ranger` function, using the parameters identified earlier.

My first random forest model (using the `randomForest` package) obtained 2.99% OOB prediction error. The confusion matrix is shown below:

```
Confusion matrix:
    B   M class.error
B 350   7  0.01960784
M  10 202  0.04716981
```

The second random forest (using the `ranger` package) obtained 3.34% OOB prediction error. The confusion matrix is shown below:

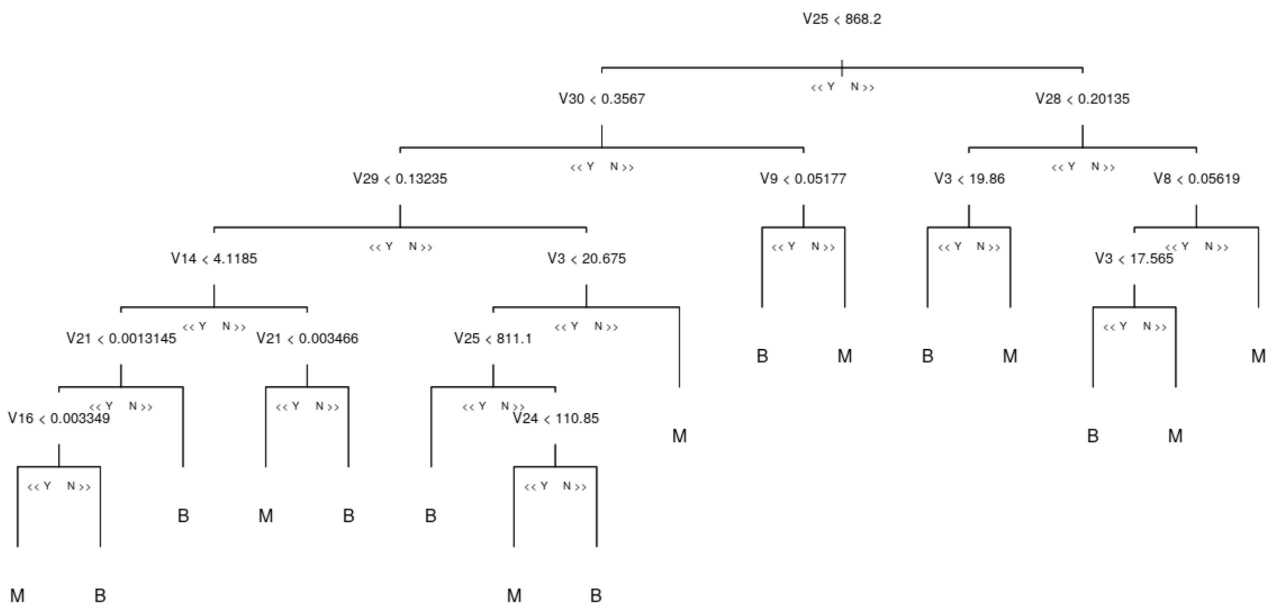
```
     1   2
B  351  13
M    6 199
```

Both implementations of random forest performed well, achieving low misclassification rates.

Finally, to create a graphical representation of the `randomForest` object I called the `reprtree` package. **Figure 13** visualizes a "representative tree" aggregated from the ensemble, with branches based on aggregated decision rules. At terminal nodes, `M` means malignant and `B` means benign tumor.

Figure 13

Dendrogram of Random Forest Model



4 Discussion / Conclusion

In this analysis, I achieved my goal of developing several classification models to classify tumors as benign or malignant. **Table 1** compares misclassification rates across implementations.

Table 1

Model	B classified as M	M classified as B	% Misclassified	Estimated Prediction Error
QDA using PC1, PC2	14	18	5.62%	5.98% Holdout Method
QDA using all data	9	5	2.46%	4.39% Holdout Method
Random Forest (from `randomForest`)	10	7	2.99%	2.99% OOB
Random Forest (from `ranger`)	6	13	3.34%	3.34% OOB

For random forests, the percent misclassified and the estimated prediction error are the same because the model optimizes OOB over many iterations of training individual trees. The inventor of random forests claimed that “unlike cross-validation, where bias is present but its extent unknown, the out-of-bag estimates are unbiased” (Breiman 2001), but this claim has been challenged (Janitza & Hornung 2018). Some authors report the OOB error has bias that is sensitive to the choice of tuning parameters . No error metric is perfect, and it can be helpful to examine a variety of best-fit measures.

Despite parametric assumptions, quadratic discriminant analysis performed very well. In particular, the QDA model trained using all available data achieved the best misclassification rate of any model (2.46%). While it is difficult to estimate true prediction error without a test data set, the E(AER) was much higher (4.39%) and provides a less optimistic error estimate. It might be useful to perform *k*-fold cross-validation to estimate error. Future research should also address how violations of assumptions distort error estimates for QDA.

In summary, my analysis reveals a concrete example of the application of QDA and random forest models to high-dimensional classification problems. In addition, I demonstrated the importance of checking assumptions, tuning model parameters, and evaluating multiple error metrics.

6 Appendix

Citations

Janitza, S., & Hornung, R. (2018). On the overestimation of random forest's out-of-bag error. *PloS one*, *13*(8), e0201904. https://doi.org/10.1371/journal.pone.0201904

Lantz, L. (2019). Evaluation of the robustness of different classifiers under low- and high-dimensional settings (Doctoral thesis). Uppsala University, Sweden. https://uu.diva-portal.org/smash/get/diva2:1325110/FULLTEXT01.pdf

Li, J. 2013. Predicting the spatial distribution of seabed gravel content using random forest, spatial interpolation methods and their hybrid methods. Pages 394-400. The International Congress on Modelling and Simulation (MODSIM) 2013, Adelaide.

Næs, Tormod & Mevik, Bjørn-Helge. (2001). Understanding the collinearity problem in regression and discriminant analysis. Journal of Chemometrics. 15. 413 - 426. 10.1002/cem.676.

Street, Nick & Wolberg, William & Mangasarian, O. (1993). Nuclear Feature Extraction For Breast Tumor Diagnosis. Proc. Soc. Photo-Opt. Inst. Eng.. 1993. 10.1117/12.148698.

Wright, M. N. & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. J Stat Softw 77:1-17. http://dx.doi.org/10.18637/jss.v077.i01.

Supplementary Table 1

Adapted directly from Street, Wolberg, and Mangasarian, 1993.

Feature	Description
1. Radius	The radius of an individual nucleus is measured by averaging the length of the radial line segments defined by the centroid of the snake and the individual snake points.
2. Perimeter	The total distance between the snake points constitutes the nuclear perimeter.
3. Area	Nuclear area is measured simply by counting the number of pixels on the interior of the snake and adding one-half of the pixels in the perimeter.
4. Compactness	Perimeter and area are combined to give a measure of the compactness of the cell nuclei using the formula perimeter^2/area . This dimensionless number is minimized by a circular disk and increases with the irregularity of the boundary. However, this measure of shape also increases for elongated cell nuclei, which do not necessarily indicate an increased likelihood of malignancy. The feature is also biased upward for small cells because of the decreased accuracy imposed by digitization of the sample. We compensate for the fact that no single shape measurement seems to capture the idea of “irregular” by employing several different shape features.
5. Smoothness	The smoothness of a nuclear contour is quantified by measuring the difference between the length of a radial line and the mean length of the lines surrounding it. This is similar to the curvature energy computation in the snakes.
6. Concavity	In a further attempt to capture shape information we measure the number and severity of concavities or indentations in a cell nucleus. We draw chords between non-adjacent snake points and measure the extent to which the actual boundary of the nucleus lies on the inside of each chord. This parameter is greatly affected by the length of these chords, as smaller chords better capture small concavities. We have chosen to emphasize small indentations, as larger shape irregularities are captured by other features.
7. Concave Points	This feature is similar to Concavity but measures only the number, rather than the magnitude, of contour concavities.
8. Symmetry	In order to measure symmetry, the major axis, or longest chord through the center, is found. We then measure the length difference between lines perpendicular to the major axis to the cell boundary in both directions. Special care is taken to account for cases where the major axis cuts the cell boundary because of a concavity.
9. Fractal Dimensions	The fractal dimension of a cell is approximated using the “coastline approximation” described by Mandelbrot. The perimeter of the nucleus is measured using increasingly larger 'rulers'. As the ruler size increases, decreasing the precision of the measurement, the observed perimeter decreases. Plotting these to values on a log scale and measuring the downward slope gives (the negative of) an approximation to the fractal dimension. As with all the shape features, a higher value corresponds to a less regular contour and thus to a higher probability of malignancy.
10. Texture	The texture of the cell nucleus is measured by finding the variance of the gray scale intensities in the component pixels.

R Code

```
# Data was downloaded Monday, March 30 from the UCI Machine Learning repository:
# https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

########################### IMPORT DATA ##################################################

setwd("/home/tim/Documents/R/STAT488Multivariate") # set working directory

# 'wdbc' = Wisconsin diagnostic breast cancer data

wdbc <- read.csv("wdbc.data", header=F)[,-1] # remove index
wdbc[,1] <- factor(wdbc[,1])
colnames(wdbc) <- paste("V", 1:31, sep="")
m <- data.matrix(wdbc) # as matrix

########################### EDA AND VISUALIZATION ########################################

# 2 kinds of tumors: 357 benign and 212 malignant
unique(m[,1])
sum(m[,1] == 1)
sum(m[,1] == 2)

# Visualize data
boxplot(m[,2:31], main="Boxplot of Independent Variables",
        names=1:30, xlab="Independent Variable", ylab="Value",
)
boxplot(scale(m[,2:31]), main="Boxplot of Scaled Independent Variables",
        names=1:30, xlab="Independent Variable", ylab="Standard Deviation",
)

# Inspect variation by class
# Need to reshape data for ggplot
library(tidyr)
library(ggplot2)
wdbc.long <- gather(wdbc, key="Variable", value="Value", V2:V31, factor_key = T)
ggplot(wdbc.long, aes(x=Variable, y=log(Value + 1e-2), fill=V1)) +
  geom_boxplot(outlier.shape = NA) + theme_bw() + scale_y_discrete(breaks=NULL) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("Separation of Features for Benign and Malignant Tumors") +
  labs(y=expression(Log[10]~Value), fill = "Class")

############# NORMALITY, MULTICOLLINEARITY, AND EQUAL VARIANCE ASSUMPTIONS ##############

# Test for a difference in mean vectors

library(ICSNP) # Hotelling's T^2 test
HotellingsT2(m[,2:31] ~ m[,1]) # Do mean vectors differ between tumor classes?

# Examine multivariate normality

library(MVN)
mvn(m[,2:31], mvnTest = "royston")

# Function to randomly generate bivariate contour plots

par(mfrow=c(2, 3))
i <- sample(2:31, size = 6)
j <- sample(c(2:31)[-i+1], size = 6)
mapply(function(x,y) mvn(m[, c(x, y)],
                         mvnTest = "royston", multivariatePlot = "contour"), i, j)
remove(i); remove(j); dev.off()

# Examine pairwise correlations
```

```r

library(corrplot)
corrplot(cor(m), main="Correlation Plot")

# Some variables are functions of other variables: multicollinearity should be high
# Fit a logistic model and calculate VIF

library(car) # vif function
lm <- glm(V1 ~ ., data = wdbc, family = "binomial", control = list(maxit = 50))
vif(lm)

# Do the variance-covariance matrix and correlation matrix differ between classes?

all.equal(cov(subset(wdbc, V1=="B")[,2:31]), cov(subset(wdbc, V1=="M")[,2:31]))
all.equal(cor(subset(wdbc, V1=="B")[,2:31]), cor(subset(wdbc, V1=="M")[,2:31]))

#################### PRINCIPAL COMPONENT ANALYSIS ########################################

pca <- prcomp(m[,2:31], retx = T, scale = T, center = T) # PCA

# Visualize cumulative proportion of variance explained by each PC

plot(cumsum(pca$sdev^2)/30, main="Scree Plot",
     ylab="Proportion of Explained Variance", xlab="Principal Component")
lines(cumsum(pca$sdev^2)/30)

# Visualize the projection of data onto PC1 and PC2

plot(pca$x[,1:2], col = c("black", "red")[m[,1]],
     main = "Separation of Malignant Tumors (Red) and Benign Tumors (Black) after PCA")

# Draw confidence ellipses

dataEllipse(x=pca$x[which(wdbc$V1 == "B"),1], y=pca$x[which(wdbc$V1 == "B"),2],
            center.cex=.75, xlim=c(-15, 6), ylim=c(-13, 9), center.pch=3,
            xlab="PC1", ylab="PC2",log="", levels=c(0.5, 0.95), col="black",
            main="0.50 and 0.95 Confidence Ellipses for Benign and Malignant Tumors")
dataEllipse(x=pca$x[which(wdbc$V1 == "M"),1], y=pca$x[which(wdbc$V1 == "M"),2],
            center.cex=.75, log="", levels=c(0.5, 0.95),
            center.pch=3,col="red", add=TRUE)
legend(-15,-4, fill=c("black","red"),legend=c("Benign Tumor","Malignant Tumor"),
       bg="transparent", cex = 0.75, bty = "n")

#################### QUADRATIC DISCRIMINANT ANALSIS ######################################
########################## USING PC1 AND PC2 ###########################################

library(MASS) # quadratic discriminant analysis (QDA)

pc.dat <- data.frame(cbind(m[,1], pca$x[,1:2])) # prepare data

qda.pca <- qda(V1~PC1+PC2, data=pc.dat, prior=c(0.5,0.5)) # QDA

CM <- table(m[,1], predict(qda.pca, pc.dat)$class); CM # confusion matrix

(sum(CM)-sum(diag(CM)))/sum(CM)*100 # APER (apparent error rate) is approx. 5.62%

holdout.class <- NULL
for (i in 1:569){
  z <- qda(V1~PC1+PC2, data=pc.dat, prior=c(0.5,0.5), subset = c(1:569)[-i])
  holdout.class[i] <- predict(z, pc.dat[i, 2:3])$class
}

CM <- table(m[,1], holdout.class); CM # confusion matrix

(sum(CM)-sum(diag(CM)))/sum(CM)*100 # Expectation of actual error rate = 5.98%
```

```

#################### QUADRATIC DISCRIMINANT ANALSIS #####################################
######################### USING ALL VARIABLES ###########################################

qda <- qda(m[,1]~m[,2:31], prior=c(0.5,0.5))

CM <- table(m[,1], predict(qda, wdbc)$class); CM # confusion matrix

(sum(CM)-sum(diag(CM)))/sum(CM)*100 # APER (apparent error rate) is approx. 2.46%

holdout.class <- NULL
for (i in 1:569){
  z <- qda(V1 ~ ., data=wdbc[,1:31], prior=c(0.5,0.5), subset = c(1:569)[-i])
  holdout.class[i] <- predict(z, wdbc[i, 2:31])$class
}

CM <- table(m[,1], holdout.class); CM # confusion matrix

(sum(CM)-sum(diag(CM)))/sum(CM)*100 # Expectation of actual error rate = 4.39%

################## PARAMETER TUNING FOR RANDOM FOREST ###################################

# Below section is computationally intensive

set.seed(1234)
library(caret) # tools for parameter tuning
library(randomForest) # random forests
library(ranger) # fast implementation of random forests
library(beepr) # noise to alert end of a long process

# Configure the train() function

ctrl <- trainControl(method='repeatedcv',                      # k-fold cross validation
                     number=5,                                               # 5 folds
                     repeats=20,                                      # 20 repetitions
                     search='grid')                                   # manual grid search

# Evaluate optimal number of features to sample at each node:
# Vary the 'mtry' parameter in randomForest and compute the error

rf_gridsearch <- train(V1 ~ .,
                       data = wdbc,
                       method = 'rf',
                       metric = 'Accuracy',
                       trControl = ctrl,
                       tuneGrid = expand.grid(.mtry = (1:20)) # mtry values to test
); beepr::beep(2)

print(rf_gridsearch) # results of grid search

plot(rf_gridsearch, xlab="Randomly selected predictors",
     ylab="Accuracy (Repeated Cross-Validation)",
     main = "Optimal Number of Features to Sample at each Node\n
     Based on 20 Repetitions of 5-Fold Cross-Validation")

# Based on 5-Fold Cross-Validation, mtry=15 has the lowest error.

# Investigate a reasonable choice for the number of trees

plot(randomForest(V1 ~ ., data = wdbc, mtry = 15, ntree = 1000),
     main="Error as a Function of Number of Trees\n
     Model Error (Black), Benign Tumors (Green), and Malignant Tumors (Red)")
legend(600, 0.12, fill=c("green", "black", "red"),
       legend=c("Error for Benign Tumors", "Model Error", "Error for Malignant Tumors"),
       bg="transparent", cex = 0.75, bty = "n")
```

```

# Verify results using the ranger() implementation and test minimum node size

tunegrid <- expand.grid(.mtry = (1:30),                 # features sampled at each node
                        .splitrule = "gini",            # gini impurity, related to entropy
                        .min.node.size = (1:5))  # minimum node size

ranger.gridsearch <- train(V1 ~ .,
                           data = wdbc,
                           method = 'ranger',
                           metric = 'Accuracy',
                           trControl = ctrl,
                           tuneGrid = tunegrid
); beepr::beep(3)

print(ranger.gridsearch) # results of grid search

plot(ranger.gridsearch,
     xlab="Number of Features Randomly Sampled at Each Node",
     ylab="Accuracy (Repeated Cross-Validation)",
     main = "Optimal Hyperparameters Based on 20 Repetitions of 5-Fold Cross-Validation")

# Based on 5-Fold Cross-Validation, mtry=10 has the lowest error and min.node.size = 2.

############################### RANDOM FOREST MODEL #####################################

# Fit random forest using randomForest()

(rf <- randomForest(V1 ~ ., data = wdbc, mtry = 15, ntree = 400)) # 2.99% OOB error
(CM <- rf$confusion[,1:2]); (sum(CM)-sum(diag(CM)))/sum(CM)*100 # 2.99% prediction error

# Fit random forest using ranger()

(rf2 <- ranger(V1 ~ ., data = wdbc, mtry = 10, num.trees = 400, min.node.size=2)) # 3.34%
(CM <- table(rf2$predictions, m[,1])); (sum(CM)-sum(diag(CM)))/sum(CM)*100 # 3.34%

########################### VISUALIZE RANDOM FOREST #####################################

# Visualize a representative tree aggregated from the random forest

library("reprtree")
avgtree <- ReprTree(rf, wdbc, metric='d2')
reprtree:::plot.reprtree(avgtree)
```