ST 537 Final Project

Shantel Ward, Rong Huang, John Clements

11/17/2020

Introduction

Breast cancer is a type of cancer that starts in the breast. Breast cancer cells usually form a tumor that can often be seen on an x-ray or felt as a lump[1]. Breast tumors are either benign (non-cancerous) or malignant (cancerous). Since cancer is life threatening, it is crucial to predict if a breast tumor is benign or malignant with the highest possible accuracy levels.

Purpose

The purpose of this statistical analysis is to examine the ways in which malignant and benign tumors differ in their characteristics. The secondary purpose is to determine if an informative lower dimensional structure of the variables exists, and use that structure to train classification models to predict whether a lump is benign or malignant. Our goal is to select the classification model that yields the highest accuracy, but also to discuss the relative costs of False Positives versus False Negatives.

Scientific Questions

Are there statistically significant differences between mean vectors for benign tumors and malignant tumors? Can we use Principal Components Analysis (PCA) to reduce the dimensionality of the data and to identify/summarize crucial variables? Which classification model yields the highest accuracy for predicting if a tumor is benign or malignant?

Data Description

The data was collected from 699 patients of Dr. William H. Wolberg at the University of Wisconsin between January 1989 and November 1991. Samples arrived periodically as Dr. Wolberg reported his clinical cases. Measurements were derived from fine needle aspirations of human breast masses and analyses were performed on the masses. Each observation is described by the following nine features:

1: Clump Thickness

2: Cell Size Uniformity

3: Cell Shape Uniformity

4: Marginal Adhesion

5: Single Epithelial Cell Size

6: Bare Nuclei

7: Bland Chromatin

8: Normal Nucleoli

9: Mitosis

The fine needle aspirates for these nine features were graded 1 to 10 at the time of sample collection, with 1 being the closest to benign and 10 the closest to malignant. Each observation has one of two possible classes: benign or malignant. In the dataset, the benign class is labeled as 2 and the malignant class is labeled as 4. 458 (65.5%) of the observations were classified as benign and 241 (34.5%) of the observations were classified as malignant. The dataset used for this analysis is the Breast Cancer Wisconsin (Original) data set sourced from UCI's Machine Learning Repository[2].

Overview

Provide overview of conclusions of the data analysis, and a short road map for the remainder of the report.

Methods: Data Analysis

The data pre-processing and statistical analysis was performed using R version 4.0.2.

```
suppressWarnings(library(tidyverse))

##### DATA CLEANING #####
# Read in data set.

# Read in the Breast Cancer data set.
bc.df <- read.table("Data/breast-cancer-wisconsin.data", sep=",")
# Add informative column names.
colnames(bc.df) = c("Id", "Clump Thickness", "Cell Size Uniformity",
                    "Cell Shape Uniformity", "Marginal Adhesion",
                    "Single Epithelial Cell Size", "Bare Nuclei",
                    "Bland Chromatin", "Normal Nucleoli", "Mitosis",
                    "Class")

# Calculate mean Bare Nuclei by Class.
mean.BareNuclei.2 <- bc.df %>%
  group_by(Class) %>%
  summarize(Mean = mean(as.numeric(`Bare Nuclei`),
                        na.rm=TRUE))%>%filter(Class == 2)
mean.BareNuclei.4 <- bc.df %>%
  group_by(Class) %>%
```

```r
  summarize(Mean = mean(as.numeric(`Bare Nuclei`),
                        na.rm=TRUE))%>%filter(Class == 4)

# Impute missing Bare Nuclei data with the mean for each Class mean.
bc.df <- bc.df %>%
  mutate(`Bare Nuclei Revised` = ifelse(`Bare Nuclei` == "?" & Class == 2,
                                        round(mean.BareNuclei.2$Mean,2),
                                        ifelse(`Bare Nuclei` == "?" & Class =
= 4,
                                               round(mean.BareNuclei.4$Mean,2
), `Bare Nuclei`))) #%>%

# Convert Bare Nuclei Revised data to the int data type.
bc.df$`Bare Nuclei Revised` <- as.integer(bc.df$`Bare Nuclei Revised`)

# Convert Class to a factor variable.
bc.df$Class = as.factor(bc.df$Class)

##### SUMMARY STATISTICS #####
bc.df.mean <- dplyr::select(bc.df, -c("Id","Bare Nuclei")) %>% group_by(Class
) %>% summarise_all("mean")
#transpose from wide to long
bc.df.mean<-as.data.frame(t(dplyr::select(bc.df.mean,-"Class")))
colnames(bc.df.mean) <- c("Benign", "Malignant")
bc.df.sd <- dplyr::select(bc.df, -c("Id","Bare Nuclei")) %>% group_by(Class)
%>% summarise_all("sd")
#transpose from wide to long
bc.df.sd<-as.data.frame(t(dplyr::select(bc.df.sd,-"Class")))
colnames(bc.df.sd) <- c("Benign", "Malignant")

##### DATA SPLITTING #####
# Set a random seed for reproducibility.
set.seed(17)
# Count the number of observations in the data set.
num.obs <- dim(bc.df)[1]
# Set the proportion of observations to be in the training data set.
prop.train <- 0.7
# Set the number of observations to be in the training data set.
num.train <- round(num.obs*prop.train)
# set the indicies of the dataset to be in the training data set.
train.indices <- sample(1:num.obs, num.train, replace=FALSE)
# Subset the data into training and testing data sets.
train.bc.df <- bc.df[train.indices,]
test.bc.df <- bc.df[-train.indices,]

# Save the vector of the indicies of the independent variable columns.
X_col_indicies <- c(2, 3, 4, 5, 6, 8, 9, 10, 12)
```

Missing Data

Upon review of the dataset, missing values were identified in the Bare Nuclei feature for 16 observations. The missing values are denoted as "?" in the dataset. The missing Bare Nuclei values were imputed with the mean of the non-missing Bare Nuclei values from the corresponding class (benign or malignant). For example, if the missing Bare Nuclei was of the benign class, then that missing value was imputed with the mean of the Bare Nuclei values that were in the benign class. A new column, Bare Nuclei Revised, was added to the data set and this new column includes the imputed values.

Summary Statistics: Mean

Table 1 shown to the right, shows the mean value for each feature by class (benign or malignant). For the benign class, each of the means fall between 1 and 3. For the malignant class, each of the means fall between 5 and 8 with the exception of Mitosis. The Mitosis mean for the malignant class is 2.59 which is far below the means for the other features in the malignant class.

```
knitr::kable(bc.df.mean, align = "c",
             caption =  "Mean Summary Statistics", digits = 2)
```

Mean Summary Statistics

	Benign	Malignant
Clump Thickness	2.96	7.20
Cell Size Uniformity	1.33	6.57
Cell Shape Uniformity	1.44	6.56
Marginal Adhesion	1.36	5.55
Single Epithelial Cell Size	2.12	5.30
Bland Chromatin	2.10	5.98
Normal Nucleoli	1.29	5.86
Mitosis	1.06	2.59
Bare Nuclei Revised	2.40	4.70

Summary Statistics: Standard Deviation

Table 2 shown to the right, shows the standard deviation value for each feature by class (benign or malignant). For the benign class, each of the standard deviations fall between 0 and 2. For the malignant class, each of the standard deviations fall between 2 and 4. This indicates that there is more variability in the malignant class when compared to the benign class.

```
knitr::kable(bc.df.sd, align = "c",
             caption =  "Standard Deviation Summary Statistics", digits = 3)
```

Standard Deviation Summary Statistics

	Benign	Malignant
Clump Thickness	1.674	2.429
Cell Size Uniformity	0.908	2.720
Cell Shape Uniformity	0.998	2.562
Marginal Adhesion	0.997	3.210
Single Epithelial Cell Size	0.917	2.452
Bland Chromatin	1.080	2.274
Normal Nucleoli	1.059	3.351
Mitosis	0.502	2.558
Bare Nuclei Revised	1.204	2.646

Data Pre-Processing

After imputing the missing Bare Nuclei values, the Bare Nuclei Revised variable was converted to an integer and the Class variable was converted to a factor with levels 2 and 4. Then the observed data was randomly split into a testing and training dataset. 70% (n = 489) of the observations were assigned to the testing dataset and 30% (n = 210) of the observations were assigned to the training dataset.

Methods: Statistical Models

Let us define the p observed features in a data vector, X, such that $X = (X_1, \ldots, X_p)^T$ where

X_1: Clump Thickness in {1,2,3,4,5,6,7,8,9,10}

X_2: Cell Size Uniformity in {1,2,3,4,5,6,7,8,9,10}

X_3: Cell Shape Uniformity in {1,2,3,4,5,6,7,8,9,10}

X_4: Marginal Adhesion in {1,2,3,4,5,6,7,8,9,10}

X_5: Single Epithelial Cell Size in {1,2,3,4,5,6,7,8,9,10}

X_6: Bland Chromatin in {1,2,3,4,5,6,7,8,9,10}

X_7: Normal Nucleoli in {1,2,3,4,5,6,7,8,9,10}

X_8: Mitosis in {1,2,3,4,5,6,7,8,9,10}

X_9: Bare Nuclei Revised in {1,2,3,4,5,6,7,8,9,10}

Hotelling's T^2 Two-Sample Test

Multiple variables were collected to characterize benign and malignant tumors, and one of our interests is to investigate whether mean values of those variables are distinct within

different tumor classes. The dataset contains measurement of p = 9 characteristic vairables of m=241 benign tumors and n=458 malignant tumors:

$X_i = (X_{1i}, X_{2i}, \ldots, X_{pi})^T$ for the i^{th} subject with benign tumors, $i = 1, \ldots, m$;

$Y_i = (Y_{1i}, Y_{2i}, \ldots, Y_{pi})^T$ for the i^{th} subject with malignant tumors, $i = 1, \ldots, n$.

Benign tumors: $X_i, \ldots X_m \sim N(\mu_1, \Sigma)$

Malignant tumors : $Y_i, \ldots Y_n \sim N(\mu_2, \Sigma)$

Two mean vectors are: $\mu_1 = E(X_i) = (\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_p)$, $\mu_2 = E(Y_i) = (\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_p)$

We want to test $H0: \mu_1 = \mu_2$ vs. $Ha: \mu_1 \neq \mu_2$. Note that we are making the following assumptions [4]:

Both populations are multivariate normal.

Both populations have the same variance-covariance matrix.

Both samples are mutually independent.

The difference 1-2 can be estimated by $\overline{X} - \overline{Y}$ as shown below:

$$\overline{X} - \overline{Y} \sim N[\mu_1 - \mu_2, \Sigma(\frac{1}{m} + \frac{1}{n})]$$

To test the hypothesis, we used a two-sample Hotelling's test. The test statistic is

$$T^2 = \frac{m+n-p-1}{(m+n-2)p}(\overline{X} - \overline{Y})^T (S_{pool}(\frac{1}{m} + \frac{1}{n}))^{-1}(\overline{X} - \overline{Y})$$

where $S_{pool} = \frac{(m-1)S_1 + (n-1)S_2}{m+n-2}$ is called the pooled covariance matrix, and it is an estimator of Σ. S_1 and S_2 are sample covariance matrices for the X and Y samples, respectively. We reject the H_0 if observed value of T^2 exceeds $F_{p,m+n-p}(\alpha)$ at $\alpha = 0.05$ significance level.

To quantify the difference between each element of $\mu_1 - \mu_2$, we constructed simultaneous confidence intervals which are called the Bonferroni intervals as shown below.

For 1k-2k: (xk-yk)tm+n-2(2p)(1m+1n)Spool, kk

Where $S_{pool,kk}$ is the k^th diagonal entry of S_{pool},and $\mu_{1k} - \mu_{2k}$ is the difference of the k^{th} variable in the mean vectors. The intervals are constructed at $100 * (1 - \frac{\alpha}{p})\%$ confidence level.

Principal Components Analysis (PCA)

The main goal of PCA is to identify linear combinations of X of the form $Y_i = a_i^T X$, where $i = 1,2, \ldots, q$,that explains most of the variability in X. Usually, $q < p$. a_i^T is a vector of length p which represents the PC loadings and tells us how the original features are weighted to get

the PCs. Y_i represents the new variables and are ordered according to their importance. For instance, Y_1 is designed to capture the most variability in the original features by any linear combination. Y_2 then captures the most of the remaining variability while being uncorrelated to Y_1, and so on. We hope that the first few Y_i"s capture most of the variability in X. If we are able to capture most of the variability in X with a few Y_i"s , then we have achieved dimensionality reduction by condensing a sufficient portion of the information present in the original set of features via linear combinations while losing as little information as possible.

We used the prcomp function from the stats package in R to conduct the PCA. The PCA was run using the training dataset with the center and scale arguments set equal to true to standardize the dataset. The results and interpretation of the PC loadings will be discussed in the results section.

Classification

Scientific Question #2: Which classification model yields the highest accuracy for predicting if a tumor is benign or malignant, provided there are statistically significant differences in the measurements taken from benign and malignant masses?

After establishing there are statistically significant differences between benign and malignant masses in the measured variable and if there was a lower dimensional representation of the data containing sufficient information, we proceeded to build predictive models using the caret package. The tested models include K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Classification Trees, and Support Vector Machines (SVM) with a Radial Basis Kernel. The performance of these models were examined using 5-fold Cross Validation repeated 20 times, for 100 total resamples from the training data set. This process was repeated for data transformed based on the results from the Principal Components Analysis. The distribution of accuracies and Cohen's κ were evaluated to select a model for evaluation on the withheld testing data set.

The KNN model finds the k closest points to a new point and predict its class with the majority class of the k closest points. LDA assumes the variables of both classes are multivariate normal distributions with different means, but the same covariance matrix. LDA predicts the class of new data points based on which class probability density distribution has the higher density for the new point. QDA works the same as as LDA, but allows for the covariance matrices of the classes to differ. Classification Trees work by finding the best split point among the independent variables, where best means improving the classification of the training data by some metrix (Gini Impurity, Information Gain, etc.). Splits are made until a new split does not improve the fit or the number of samples at an end node is too small. Predictions are made by majority vote of the observations at an end node. SVMs work to find a seperating hyperplane between the classes; the flexibility of the seperating hyperplane is determined by a kernel function.

The following hyperparameters were evaluated for each model on both the raw training data and a subset of the PCA scores of the training data.

KNN: k in {1,3,5,7,9}

LDA: None

QDA: None

Classification Tree: *cp* in {0.05,0.1,0.15, … ,0.5}

SVM: C in {0.25,0.5,1,2,4} and Sigma in {0.125,0.25,0.5,1,2,4,8}

The k hyperparameter for the KNN model is the number of points nearest to the new data point. Their majority vote decides the predicted class of the new data point.

The *cp* hyperparameter for the classification tree model determines the minimum decrease in the lack of fit of the model for the model to create an additional split.

The SVM has two hyperparameters: C and Sigma. C controls the number and severity of incorrect classifications. Sigma is the "inverse kernel width for the Radial Basis kernel function" (https://www.rdocumentation.org/packages/kernlab/versions/0.9-29/topics/ksvm).

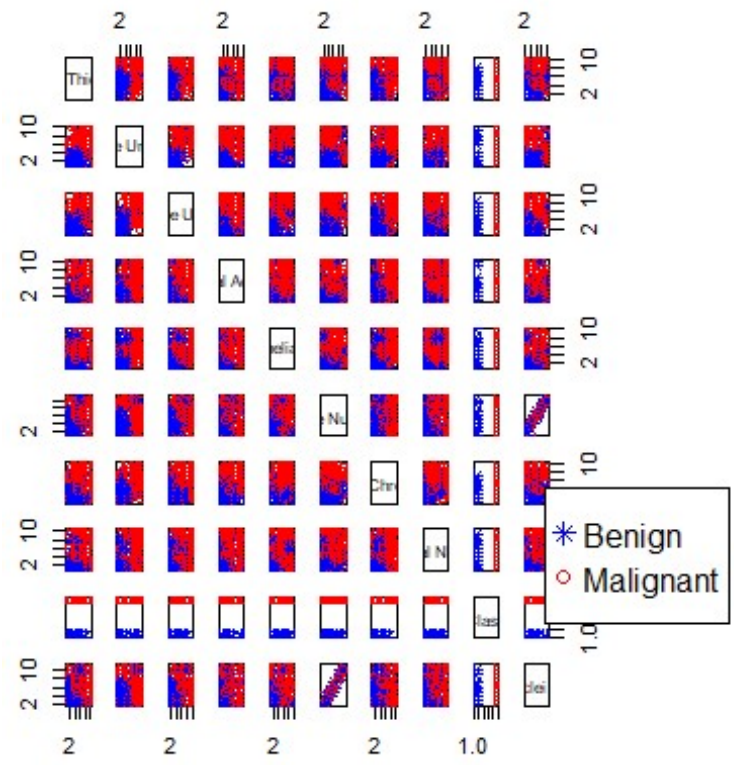
After choosing the model(s) with the best results from the cross-validation, the model(s) were tested on the testing data set and the Accuracy, Apparent Error Rate (APER), sensitivity, and specificity were examined, as well as the costs of False Positives and False Negatives.

Methods: Results

Hotelling's T2 Two Sample Test Results

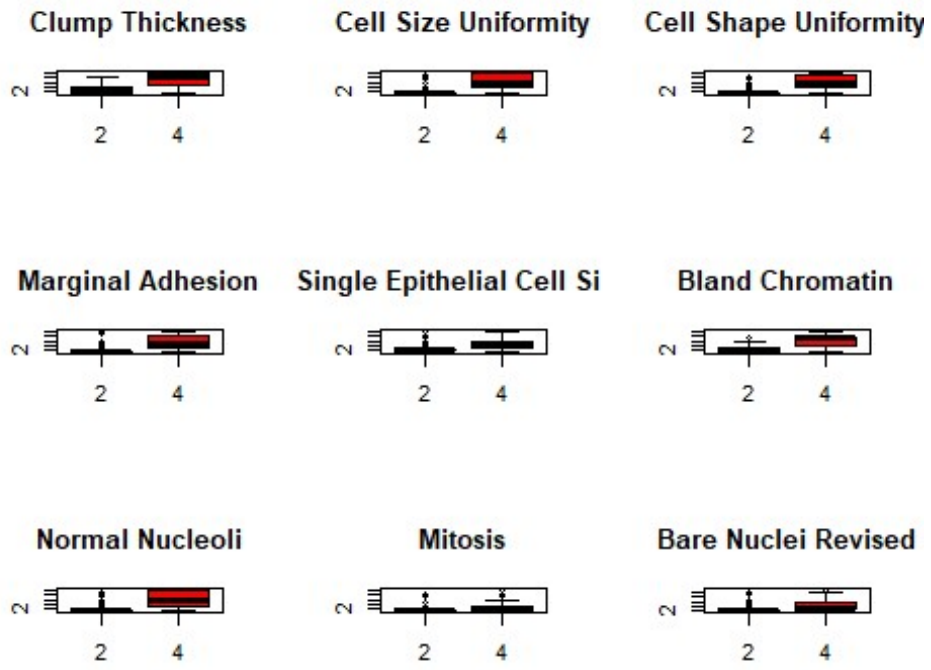
Scientific Question #1: Are there statistically significant differences between mean vectors for benign tumors and malignant tumors?

```
# pair plot
pairs(bc.df[,-c(1,10)],
      col = c(4, 2)[as.numeric(bc.df$Class)],   # Change color by group
      pch = c(8, 1)[as.numeric(bc.df$Class)],   # Change points by group
      main = "Pair Plot of Benign and Malignant for All Variables", oma=c(2,2
,2,16))
par(xpd = TRUE)
legend("bottomright", col=c(4, 2),pch = c(8, 1),
       legend = c("Benign", "Malignant"))
```



The boxplots above were depicted for each variable for both benign and malignant tumors. As domenstrated in the graph, the benign tumors have more outliers compared to malignant tumors. Moreover, the values of each variable for benign seem to be smaller than those of malignant.

```
# boxplot
par(mfrow=c(3,3))
for(col_index in X_col_indicies){
  boxplot(bc.df[,col_index] ~ Class, data = bc.df, col=c(4,2),
          xlab="", ylab = "", main=names(bc.df)[col_index])
}
```



The chi-square plots in Figure 2 were created to validate the assumption of normality. If the multivariate normality assumption is correct, the points should follow a straight line. It is true for the malignant class, but a curve is detected for the benign class which suggests a departure from normality. The two points that show the largest deviation from the linear trend are highlighted in red circles. But considering the sample sizes ($m = 241; n = 458$) are large enough, Hotelling's T^2 test is still quite robust even if there are slight departures from normality and several outliers are observed.

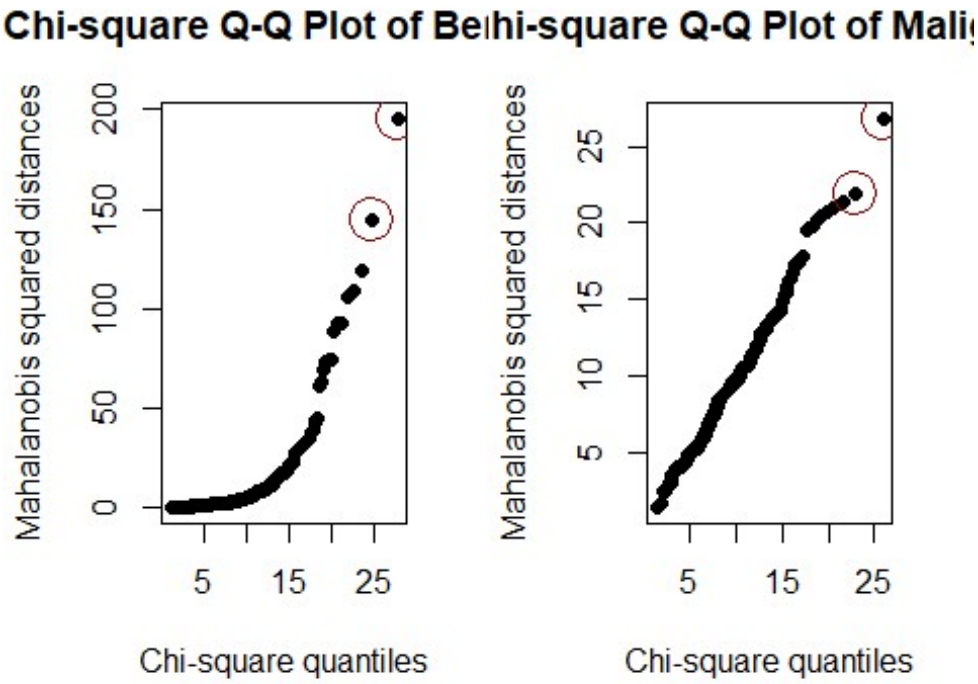
Below presents the result from Hotelling's T^2 test. The small p-value ($< 2.2e - 16$) suggests we reject the null hypothesis and concludes that there is a difference in mean vector between benign and malignant tumors.

```
suppressWarnings(library(ICSNP))

# Seperate the 2 classes into dataframes for mean vector comparison.
X = as.matrix(bc.df[bc.df$Class == "2", X_col_indicies]) # class 2
Y = as.matrix(bc.df[bc.df$Class == "4", X_col_indicies]) # class 4

#normality check
################## chisquare.plot ##############
chisquare.plot <- function(x, mark, title) {
  # x = A nxp data matrix, mark: number of extreme points to mark,
  # p = number of variables, n = sample size
  p <- ncol(x)
  n <- nrow(x)
  # xbar and s
```

```
  xbar <- colMeans(x)
  s <- cov(x)
  # Mahalanobis dist, sorted
  x.cen <- scale(x, center = T, scale = F)
  d2 <- diag(x.cen %*% solve(s) %*% t(x.cen))
  sortd <- sort(d2)
  # chi-sq quantiles
  qchi <- qchisq((1:n - 0.5)/n, df = p)
  # plot, mark points with heighest distance
  plot(qchi, sortd, pch = 19, xlab = "Chi-square quantiles",
       ylab = "Mahalanobis squared distances",
       main = title)
  points(qchi[(n - mark + 1):n], sortd[(n - mark + 1):n], cex = 3, col = "#99
0000")
}
###############################################
par(mfrow=c(1,2))
chisquare.plot(X,2, "Chi-square Q-Q Plot of Benign")
chisquare.plot(Y,2, "Chi-square Q-Q Plot of Malignant")
```




```
## Two-sample Hotelling"s T2 test
HotellingsT2(X,Y)

##
##  Hotelling's two sample T2-test
##
## data:  X and Y
```

```
## T.2 = 312.56, df1 = 9, df2 = 689, p-value < 2.2e-16
## alternative hypothesis: true location difference is not equal to c(0,0,0,0
,0,0,0,0,0)
```

```
## Bonferroni intervals
alpha = 0.05 # old significance level
p = 2 # number of intervals/variables
# mean vectors
xbar = colMeans(X)
ybar = colMeans(Y)
difference = xbar - ybar
# covariances of each group
S.x = cov(X)
S.y = cov(Y)
# bc.df statistics summary
stats = round(cbind(xbar, ybar, sqrt(diag(S.x)), sqrt(diag(S.y))),3)
colnames(stats) = c("Benign Mean", "Malignant Mean", "Benign Sd", "Malignant
Sd")
stats
```

```
##                             Benign Mean Malignant Mean Benign Sd Malignant
Sd
## Clump Thickness                   2.956          7.195     1.674        2.
429
## Cell Size Uniformity              1.325          6.573     0.908        2.
720
## Cell Shape Uniformity             1.443          6.560     0.998        2.
562
## Marginal Adhesion                 1.365          5.548     0.997        3.
210
## Single Epithelial Cell Size       2.120          5.299     0.917        2.
452
## Bland Chromatin                   2.100          5.979     1.080        2.
274
## Normal Nucleoli                   1.290          5.863     1.059        3.
351
## Mitosis                           1.063          2.589     0.502        2.
558
## Bare Nuclei Revised               2.402          4.701     1.204        2.
646
```

```
# sample sizes
m = nrow(X)
n = nrow(Y)
# pooled covariance matrix
S.pool = ((m-1)*S.x + (n-1)*S.y) / (m + n - 2)
# critical value
crit = qt(alpha/(2*p), df = m+n-2, lower.tail = F)
```

The 95% Bonferroni intervals for mean difference (1-2) are displayed below. Since the difference show the negative signs all the time, it indicates that the values of each variable for benign tumors are significantly lower than those for malignant tumors.

```
# Bonferroni intervals
half.width = crit*sqrt(diag(S.pool)*(1/m + 1/n))
lower = difference - half.width
upper = difference + half.width
int.bonf = cbind(lower, upper)
int.bonf
```

```
##                                lower     upper
## Clump Thickness             -4.590312 -3.887066
## Cell Size Uniformity        -5.561346 -4.933228
## Cell Shape Uniformity       -5.422027 -4.811842
## Marginal Adhesion           -4.549452 -3.816726
## Single Epithelial Cell Size -3.468066 -2.889270
## Bland Chromatin             -4.164017 -3.593616
## Normal Nucleoli             -4.956102 -4.189253
## Mitosis                     -1.803867 -1.247919
## Bare Nuclei Revised         -2.627205 -1.971791
```

Principal Components Analysis (PCA) Results

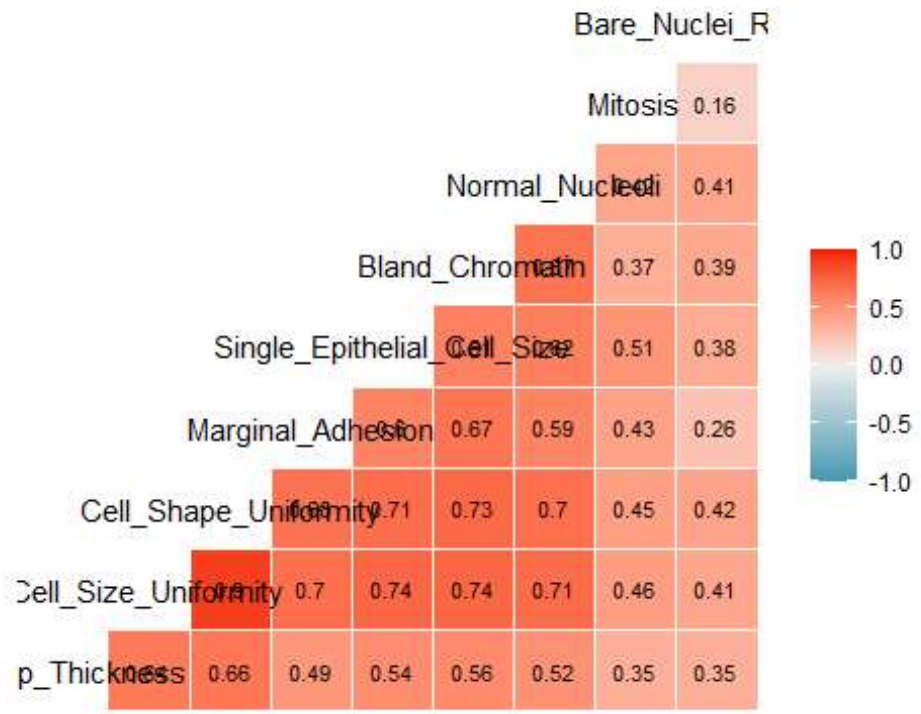
Scientific Question #2: Can we use PCA to reduce the dimensionality of the data and to identify/summarize crucial variables?

The results of the PCA importance of components are displayed below in Table 3. The output shows that 70.7% of the total variation in the training data can be explained by PC1 and PC2. We typically retain PCs that explain between 70% and 95% of the total variation.

```
##### PCA (Principle Components Analysis #####
# Scientific Question 2
# Can we use PCA to reduce the dimensionality of the data and
# to identify/summarize crucial variables?
suppressWarnings(library(knitr))
suppressWarnings(library(GGally))

# Split the training and testing data sets into independent
# variables and dependent variables.
train.bc.df.class <- train.bc.df$Class
train.bc.df.X <- select(train.bc.df, -c("Id","Bare Nuclei","Class"))
test.bc.df.class <- test.bc.df$Class
test.bc.df.X <- select(test.bc.df, -c("Id","Bare Nuclei","Class"))

# Plot a correlation matrix.
ggcorr(train.bc.df.X, label=TRUE, label_size=3, label_round=2)
```




```
# Shows that Cell Shape Uniformity and Cell Size Uniformity are highly correl
ated (0.90).
cor(train.bc.df.X)
```

```
##                              Clump Thickness Cell Size Uniformity
## Clump Thickness                    1.0000000            0.6429847
## Cell Size Uniformity               0.6429847            1.0000000
## Cell Shape Uniformity              0.6631989            0.9049778
## Marginal Adhesion                  0.4912901            0.6986296
## Single Epithelial Cell Size        0.5421161            0.7391602
## Bland Chromatin                    0.5577657            0.7431113
## Normal Nucleoli                    0.5235813            0.7106115
## Mitosis                            0.3544657            0.4570546
## Bare Nuclei Revised                0.3544358            0.4080672
##                              Cell Shape Uniformity Marginal Adhesion
## Clump Thickness                          0.6631989         0.4912901
## Cell Size Uniformity                     0.9049778         0.6986296
## Cell Shape Uniformity                    1.0000000         0.6841941
## Marginal Adhesion                        0.6841941         1.0000000
## Single Epithelial Cell Size              0.7061165         0.5979035
## Bland Chromatin                          0.7288692         0.6722778
## Normal Nucleoli                          0.6992053         0.5905170
## Mitosis                                  0.4466957         0.4325498
## Bare Nuclei Revised                      0.4203279         0.2554196
##                              Single Epithelial Cell Size Bland Chromatin
## Clump Thickness                                0.5421161       0.5577657
## Cell Size Uniformity                           0.7391602       0.7431113
```

```
## Cell Shape Uniformity                          0.7061165       0.7288692
## Marginal Adhesion                              0.5979035       0.6722778
## Single Epithelial Cell Size                    1.0000000       0.6104333
## Bland Chromatin                                0.6104333       1.0000000
## Normal Nucleoli                                0.6169662       0.6729169
## Mitosis                                        0.5081960       0.3678463
## Bare Nuclei Revised                            0.3767608       0.3948828
##                               Normal Nucleoli   Mitosis Bare Nuclei Revised
## Clump Thickness                     0.5235813 0.3544657           0.3544358
## Cell Size Uniformity                0.7106115 0.4570546           0.4080672
## Cell Shape Uniformity               0.6992053 0.4466957           0.4203279
## Marginal Adhesion                   0.5905170 0.4325498           0.2554196
## Single Epithelial Cell Size         0.6169662 0.5081960           0.3767608
## Bland Chromatin                     0.6729169 0.3678463           0.3948828
## Normal Nucleoli                     1.0000000 0.4185491           0.4142987
## Mitosis                             0.4185491 1.0000000           0.1649736
## Bare Nuclei Revised                 0.4142987 0.1649736           1.0000000
```

```
# Standardize the variables.
train.bc.df.X.std <- scale(train.bc.df.X, center=TRUE, scale=TRUE)

# Perform PCA.
data.pca <- prcomp(train.bc.df.X.std)

# Extract the importance of each component.

# standard deviation
st.dev <- data.pca$sdev
# variance
var <- st.dev^2
# total variance
TV <- sum(var)
#proportion of variance explained
prop <- var/TV
#cumulative proportion of variance explained
pve <- cumsum(var)/TV
#combine in a table
tab <- rbind(st.dev, prop, pve)
rownames(tab) <- c("Standard deviation", "Proportion of variance",
                   "Cumulative proportion")
colnames(tab) <- paste0("PC",1:9)
knitr::kable(tab, align = "c",
             caption = "PCA Importance of Components", digits = 3)
```

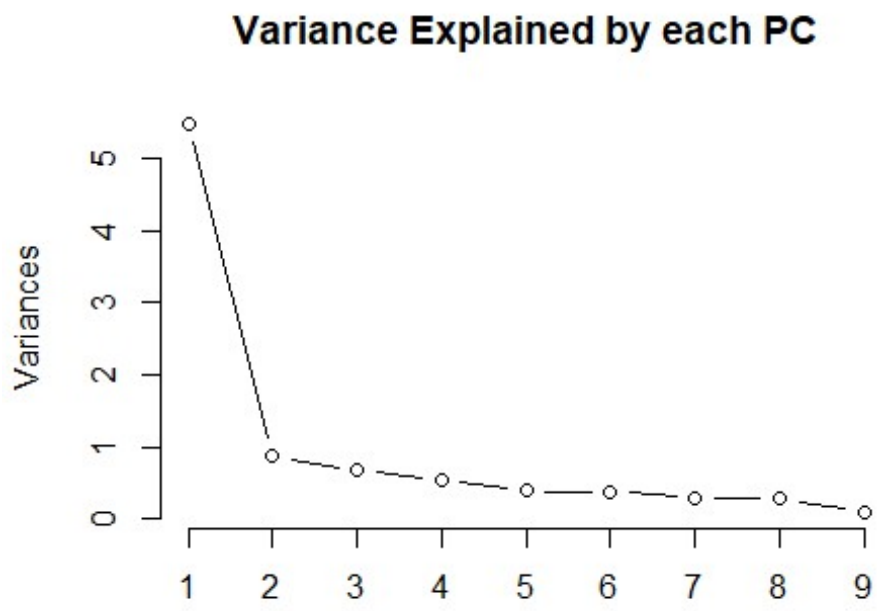
PCA Importance of Components

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Standard deviation	2.343	0.935	0.822	0.729	0.620	0.616	0.538	0.530	0.304

Proportion of variance	0.610	0.097	0.075	0.059	0.043	0.042	0.032	0.031	0.010
Cumulative proportion	0.610	0.707	0.782	0.841	0.884	0.926	0.958	0.990	1.000

The scree plot depicted in Figure 1 on the right displays the amount of variance explained for each PC. The scree plot shows a bend at PC2 which indicates that we should retain at least PC1 and PC2. Based on the cumulative proportion of variance explained and inspection of the scree plot, we will retain PC1 and PC2 for further analysis.

```
# Make a Scree Plot to help determine how many PCs to retain.
screeplot(data.pca, type = "line", main = "Variance Explained by each PC")
```



The loadings of the first two PCs are shown below in Table 4.

```
# The elbow in the graph is at PC2.
# PC 1 and 2 explain 74% of the variability
# Display the loadings for PC1 and PC2 for review.
#round(data.pca$rotation[, 1:2], 3)
knitr::kable(data.pca$rotation[, 1:2], align = "c",
             caption = "Loadings of the First Two PCs", digits = 3)
```

Loadings of the First Two PCs

	PC1	PC2
Clump Thickness	0.315	0.095

Cell Size Uniformity	0.393	-0.008
Cell Shape Uniformity	0.390	0.021
Marginal Adhesion	0.337	-0.218
Single Epithelial Cell Size	0.351	-0.105
Bland Chromatin	0.357	0.066
Normal Nucleoli	0.349	0.066
Mitosis	0.246	-0.570
Bare Nuclei Revised	0.218	0.773

Hence the first PC can be constructed as follows: $Y_1 = 0.306X_1 + 0.380X_2 + 0.377X_3 + 0.334X_4 + 0.337X_5 + 0.347X_6 + 0.332X_7 + 0.235X_8 + 0.330X_9$

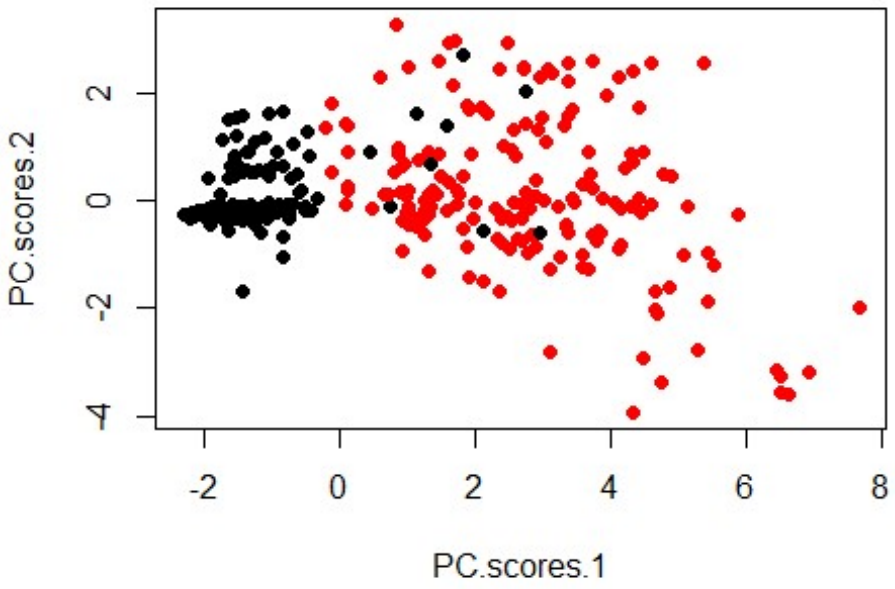
We can interpret the first PC as a roughly equally weighted sum of the nine features.

Hence the second PC can be constructed as follows: $Y_2 = 0.164X_1 + 0.058X_2 + 0.087X_3 + 0.039X_4 - 0.217X_5 + 0.201X_6 - 0.003X_7 - 0.889X_8 + 0.288X_9$

We can interpret the second PC as a contrast between Mitosis and Single Epithelial Cell Size with Clump Thickness, Bland Chromatin, and Bare Nuclei Revised. The PC2 loadings for the other four features (Cell Size Uniformity, Cell Shape Uniformity, Marginal Adhesion, and Normal Nucleoli) are close to zero and are not important in PC2. Although several features are important for PC2, the second PC is strongly weighted by Mitosis.

Y_1 and Y_2 represent the PC scores which are computed by multiplying the standardized training dataset by their respective PC loadings. The PC scores are then used as the predictors in the classification models. Figure 2 displays a scatter plot of PC1 scores and PC2 scores colored by class (black represents benign and red represents malignant).

```
# PC1 is roughly equally weighted.
# PC2 is essentially the effect of Mitosis.
PC1 <- data.pca$rotation[, 1]
PC2 <- data.pca$rotation[, 2]
PC.scores.1 <- train.bc.df.X.std %*% PC1
PC.scores.2 <- train.bc.df.X.std %*% PC2
# Plot PC score 1 vs PC score 2 colored by class.
plot(PC.scores.1,PC.scores.2, pch=19, col=train.bc.df.class)
```



Classification

Scientific Question #3: Which classification model yields the highest accuracy for predicting if a tumor is benign or malignant?

To assess which classifier was best out of the ones specified in the methods portion, we performed 5-fold cross validation and repeated the process 20 times for 100 total resamples for each of the models on the untransformed data. We repeated this for the training data transformed by the first two principal components. We then looked at the distributions of the cross-validation accuracies and Cohen's κs, a metric that accounts for class imbalances in the data (values close to 1 are good), selecting the model(s) with the highest, and most consistent performance.

As a benchmark for all the models, we calculated the no information rate. This is the accuracy from predicting the most common class for all observations. The model accuracy must have a higher accuracy than the no information rate to be useful. The no information rate in the training data set is about 64.213%. It is about 68.571% in the testing data set.

```
# Create a dataframe only holding the X variables and the Class, starting wit
h the X variables.
train.df = train.bc.df.X
test.df = test.bc.df.X
# Re-attach the Class to the independent variables of the training and testin
g dataframes.
train.df$Class <- train.bc.df.class
test.df$Class <- test.bc.df.class
```

```r
# This is essential to get rpart to run on this data.
colnames(train.df) <- make.names(colnames(train.df))
colnames(test.df) <- make.names(colnames(train.df))

# Set the number of principal components to keep.
num.keep <- 2

# Re-fit prcomp() so it knows to center and scale input data.
pca.fit <- prcomp(train.bc.df.X, center=TRUE, scale=TRUE)

# Create a new dataframe to hold the prcomp() transformed training data.
train.pca <- as.data.frame(pca.fit$x[, 1:num.keep])
# Create a new dataframe to hold the prcomp() transformed testing data.
test.pca <- as.data.frame(predict(pca.fit, test.bc.df.X)[, 1:num.keep])

# Re-attach the Class to the PCA-trainsformed training and testing dataframes
.
train.pca$Class <- train.bc.df.class
test.pca$Class <- test.bc.df.class

# Calculate the no information rates.
no.info.rate.train <- sum(train.pca$Class == 2) / length(train.pca$Class)
no.info.rate.test <- sum(test.pca$Class == 2) / length(test.pca$Class)

print(paste("No Information Rate (Training Set): ",
            round(100*no.info.rate.train, 3), "%", sep=""))

## [1] "No Information Rate (Training Set): 64.213%"

print(paste("No Information Rate (Testing Set): ",
            round(100*no.info.rate.test, 3), "%", sep=""))

## [1] "No Information Rate (Testing Set): 68.571%"

suppressWarnings(library(caret))

suppressWarnings(library(kernlab))

# We will be using 5-fold cross-validation, repeating the process 20 times.
TrControl <- trainControl(method = "repeatedcv",
                          number = 5,
                          repeats = 20)

# Fit a KNN classifier by selecting the model with the best number of neighbo
rs
# by performing 5-fold cross-validation tuning the k parameter using the odd
numbers
# from 1 up to 9.
knn.model <- train(Class ~ ., data = train.df,
                   method = "knn",
                   trControl = TrControl,
```

```
                    tuneGrid = expand.grid(k = seq(1, 9, by=2)))

# Perform 5-fold cross-validation using a LDA classifier.
lda.model <- train(Class ~ ., data = , train.df,
                   method = "lda",
                   trControl = TrControl)

# Perform 5-fold cross-validation using a QDA classifier.
qda.model <- train(Class ~ ., data = , train.df,
                   method = "qda",
                   trControl = TrControl)

# Fit a tree classifier by selecting the model with the best complexity param
eter (cp)
# by performing 5-fold cross-validation tuning the cp parameter using 0.001,
0.01,
# and 0.1
tree.model <- train(Class ~ ., data = train.df,
                    method = "rpart",
                    trControl = TrControl,
                    tuneGrid = expand.grid(cp = seq(0.05, 0.5, by=0.05)))

# Fit a SVM classifier with a radial kernel by selecting the model tuning the
C and
# sigma parameters by performing 5-fold cross-validation tuning
# the C parameter from 0.25 to 4, doubling each time, and the sigma parameter
from
# 0.125 to 8, doubling each time
svm.model <- train(Class ~ ., data = train.df,
                   method = "svmRadial",
                   trControl = TrControl,
                   tuneGrid = expand.grid(C = 2^c(-2:2),
                                          sigma = 2^(-3:3)))

# Summarize the Accuracy and Cohen"s Kappa for each model in the 5-fold CV.
resamp <- resamples(list(KNN = knn.model, LDA = lda.model, QDA = qda.model,
                         TREE = tree.model, SVM = svm.model))

quart.1st <- function(x){
  # This helper function finds the first quartile.
  return(quantile(x, probs=0.25))
}
quart.3rd <- function(x){
  # This helper function finds the third quartile.
  return(quantile(x, probs=0.75))
}
resample.summary <- function(resample, ordered.model.names, accuracy=TRUE){
  # This function takes in a resample object, a vector of the models tested
  # in order, and a boolean for accuracy. If accuracy is true, it returns
  # a dataframe summary of the resampling accuracy for each model. Else,
```

```r
  # it returns a dataframe summary of the resampling Cohen's kappa for
  # each model.

  if(accuracy == TRUE){
    # Summarize the accuracies.
    X = resample$values[,seq(2, dim(resample$values)[2], 2)]
  }
  else{
    # Summarize the Cohen's Kappas.
    X = resample$values[,seq(3, dim(resample$values)[2], 2)]
  }
  my.min <- apply(X,MARGIN=2, FUN=min)
  my.1st <- apply(X, 2, quart.1st)
  my.median <- apply(X, MARGIN=2, FUN=median)
  my.mean <- apply(X, MARGIN=2, FUN=mean)
  my.3rd <- apply(X, 2, quart.3rd)
  my.max <- apply(X, MARGIN=2, FUN=max)

  my.summary <- matrix(c(my.min,
                         my.1st,
                         my.median,
                         my.mean,
                         my.3rd,
                         my.max),
                       nrow=(dim(resample$values)[2]-1)/2)
  my.summary <- as.data.frame(my.summary)
  colnames(my.summary) <- c("Min.", "1st Quartile", "Median",
                            "Mean", "3rd Quartile", "Max.")
  rownames(my.summary) <- ordered.model.names
  return(my.summary)
}

# Get summaries of the resampling perfomances for the models trained on the r
aw
# data.
ordered.model.names <- c("KNN", "LDA", "QDA", "TREE", "SVM")
raw.acc <- resample.summary(resamp, ordered.model.names, accuracy=TRUE)
raw.kappa <- resample.summary(resamp, ordered.model.names, accuracy=FALSE)

# Fit a KNN classifier by selecting the model with the best number of neighbo
rs
# by performing 5-fold cross-validation tuning the k parameter using the odd
numbers
# from 1 up to 9.
knn.model.pca <- train(Class ~ ., data = train.pca,
                       method = "knn",
                       trControl = TrControl,
                       tuneGrid = expand.grid(k = seq(1, 9, by=2)))

# Perform 5-fold cross-validation using a LDA classifier.
```

```
lda.model.pca <- train(Class ~ ., data = , train.pca,
                       method = "lda",
                       trControl = TrControl)

# Perform 5-fold cross-validation using a QDA classifier.
qda.model.pca <- train(Class ~ ., data = , train.pca,
                       method = "qda",
                       trControl = TrControl)

# Fit a tree classifier by selecting the model with the best complexity param
eter (cp)
# by performing 5-fold cross-validation tuning the cp parameter using 0.001,
0.01,
# and 0.1
tree.model.pca <- train(Class ~ ., data = train.pca,
                        method = "rpart",
                        trControl = TrControl,
                        tuneGrid = expand.grid(cp = seq(0.05, 0.5, by=0.05)))

# Fit a SVM classifier with a radial kernel by selecting the model tuning the
C and
# sigma parameters by performing 5-fold cross-validation tuning
# the C parameter from 0.25 to 4, doubling each time, and the sigma parameter
from
# 0.125 to 8, doubling each time
svm.model.pca <- train(Class ~ ., data = train.pca,
                       method = "svmRadial",
                       trControl = TrControl,
                       tuneGrid = expand.grid(C = 2^c(-2:2),
                                              sigma = 2^(-3:3)))

# Summarize the Accuracy and Cohen"s Kappa for each model in the 5-fold CV.
resamp.pca <- resamples(list(KNN = knn.model.pca, LDA = lda.model.pca, QDA =
qda.model.pca,
                         TREE = tree.model.pca, SVM = svm.model.pca))

# Get summaries of the resampling perfomances for the models trained on the P
CA
# transformed data.
pca.acc <- resample.summary(resamp.pca, ordered.model.names, accuracy=TRUE)
pca.kappa <- resample.summary(resamp.pca, ordered.model.names, accuracy=FALSE
)
```

In the resampling, all of the models significantly outperformed the no information rate. For all of the model specifications, the ones trained on the data transformed by the first two principal components consistently outperformed their counterparts trained on the raw training data. From the models trained on the PCA transformed data, the KNN ($k = 3$), the Classification Tree ($cp = 0.5$), and the SVM ($C = 1, Sigma = 4$) had the best performances in Accuracy and Cohen's κ. The KNN and Classification Tree slightly outperformed the SVM

and are simpler models, so we proceeded with them for testing on the withheld testing data set.

```
knitr::kable(raw.acc, align = "c",
             caption =  "Resampling Accuracy (Raw Data)", digits = 7)
```

Resampling Accuracy (Raw Data)

	Min.	1st Quartile	Median	Mean	3rd Quartile	Max.
KNN	0.9175258	0.9489796	0.9690722	0.9644172	0.9795918	1.0000000
LDA	0.8979592	0.9387755	0.9489796	0.9494887	0.9690722	0.9897959
QDA	0.8979592	0.9387755	0.9591837	0.9562424	0.9693878	1.0000000
TREE	0.8673469	0.9081633	0.9285714	0.9256701	0.9387755	0.9693878
SVM	0.9183673	0.9591837	0.9693878	0.9703471	0.9795918	1.0000000

```
knitr::kable(pca.acc, align = "c",
             caption =  "Resampling Accuracy (1st 2 PCs)", digits = 7)
```

Resampling Accuracy (1st 2 PCs)

	Min.	1st Quartile	Median	Mean	3rd Quartile	Max.
KNN	0.9489796	0.9768830	0.9795918	0.9805702	0.9897959	1
LDA	0.9175258	0.9484536	0.9589733	0.9565443	0.9693878	1
QDA	0.8979592	0.9489796	0.9591837	0.9574679	0.9693878	1
TREE	0.9489796	0.9693878	0.9795918	0.9806785	0.9897959	1
SVM	0.9387755	0.9591837	0.9793814	0.9754576	0.9897959	1

```
knitr::kable(raw.kappa, align = "c",
             caption =  "Resampling Cohen's Kappa (Raw Data)", digits = 7)
```

Resampling Cohen's Kappa (Raw Data)

	Min.	1st Quartile	Median	Mean	3rd Quartile	Max.
KNN	0.8166352	0.8919722	0.9322842	0.9227766	0.9549839	1.0000000
LDA	0.7719870	0.8640604	0.8885593	0.8889319	0.9321615	0.9779180
QDA	0.7910448	0.8715596	0.9122257	0.9072393	0.9345794	1.0000000
TREE	0.7234043	0.8046217	0.8444382	0.8409821	0.8699690	0.9345794
SVM	0.8222222	0.9122257	0.9345794	0.9360928	0.9561129	1.0000000

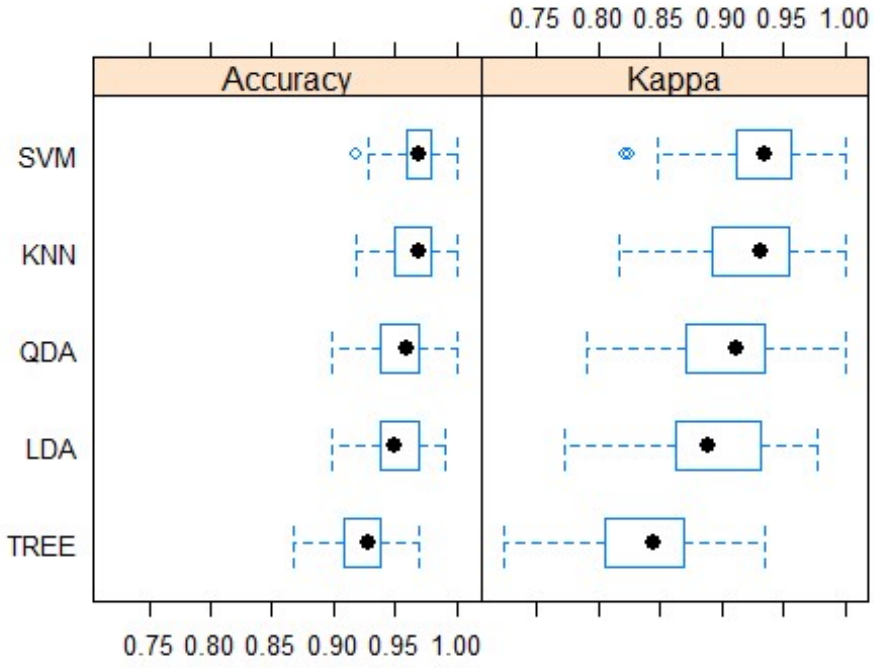
```
knitr::kable(pca.kappa, align = "c",
             caption =  "Resampling Cohen's Kappa (1st 2 PCs)", digits = 7)
```

Resampling Cohen's Kappa (1st 2 PCs)

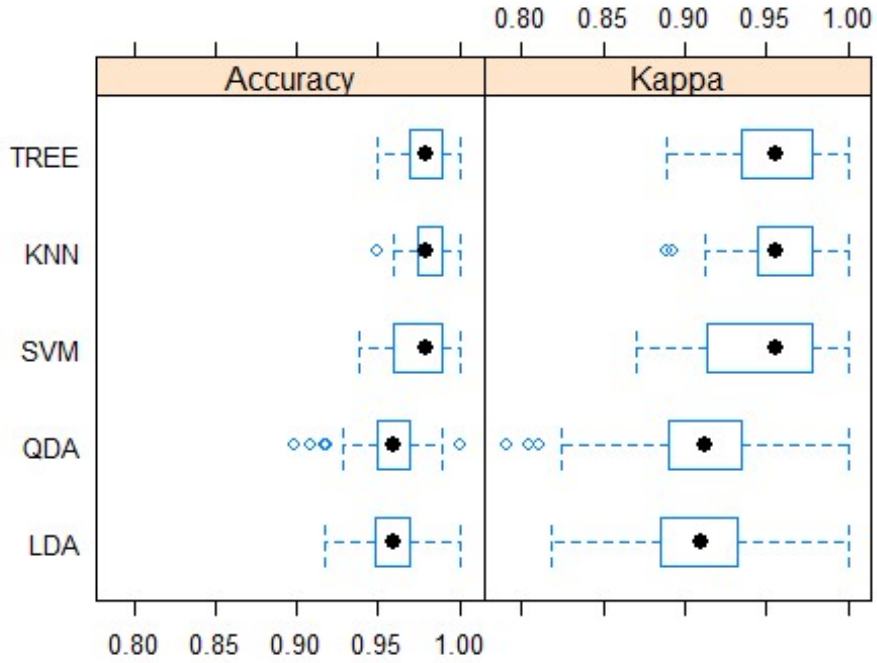
	Min.	1st Quartile	Median	Mean	3rd Quartile	Max.
KNN	0.8881789	0.9503115	0.9561129	0.9583259	0.9779180	1

LDA	0.8175896	0.8852459	0.9093813	0.9045515	0.9320388	1
QDA	0.7910448	0.8895899	0.9127698	0.9088872	0.9342782	1
TREE	0.8881789	0.9345794	0.9561129	0.9585903	0.9779180	1
SVM	0.8699690	0.9133127	0.9555556	0.9472229	0.9777880	1

```
# Make a boxplot of the model Accuracies and Cohen"s Kappas.
bwplot(resamp)
```




```
bwplot(resamp.pca)
```



We transformed the withheld testing data set with the prcomp() object fit to the training data to prevent data leakage. The KNN model and Classification Tree had nearly identical performances in the repeated 5-fold cross-validation and had identical performances on the testing data as shown in the confusion matrices below.

```
suppressWarnings(library(klaR))

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

# Get the best hyperparameter for the tree model.
best.cp = tree.model.pca$bestTune$cp
# Get predictions for the test set from the tree model.
tree.preds <- predict(tree.model.pca, test.pca)
# Make a confusion matrix for the tree model.
tree.emat <- errormatrix(test.pca$Class, tree.preds,
                         relative = TRUE)
# Convert the confusion matrix for the tree model to a dataframe.
tree.emat <- as.data.frame(round(tree.emat, 3))
# Add informative row and column names.
colnames(tree.emat) <- c('Predicted Benign', "Predicted Malignant", "Sum")
```

```
rownames(tree.emat) <- c('True Benign', "True Malignant", "Sum")

knitr::kable(tree.emat, align = "c",
             caption =  paste("Classification Tree (cp = ", best.cp,
                              ") Test Set Confusion Matrix"))
```

Classification Tree (cp = 0.5) Test Set Confusion Matrix

	Predicted Benign	Predicted Malignant	Sum
True Benign	0.938	0.062	0.062
True Malignant	0.015	0.985	0.015
Sum	0.100	0.900	0.048

```
# Get the best hyperparameter for the knn model.
best.k = knn.model.pca$bestTune$k

# Get predictions for the test set from the knn model.
knn.preds <- predict(knn.model.pca, test.pca)
# Make a confusion matrix for the knn model.
knn.emat <- errormatrix(test.pca$Class, knn.preds,
                        relative = TRUE)
# Convert the confusion matrix for the tree model to a dataframe.
knn.emat <- as.data.frame(round(knn.emat, 3))
# Add informative row and column names.
colnames(knn.emat) <- c('Predicted Benign', "Predicted Malignant", "Sum")
rownames(knn.emat) <- c('True Benign', "True Malignant", "Sum")

knitr::kable(knn.emat, align = "c",
             caption =  paste("KNN (k = ", best.k, ") Test Set Confusion Matr
ix"))
```

KNN (k = 3) Test Set Confusion Matrix

	Predicted Benign	Predicted Malignant	Sum
True Benign	0.938	0.062	0.062
True Malignant	0.015	0.985	0.015
Sum	0.100	0.900	0.048

```
# Calculate the accuracy for the tree model.
tree.accuracy <- sum(tree.preds == test.pca$Class) / length(tree.preds)
# Calculate the APER for the tree model.
tree.aper <- sum(tree.preds != test.pca$Class) / length(tree.preds)
# Calculate the sensistivity for the tree model.
tree.true.pos <- test.pca[test.pca$Class == 4, "Class"]
tree.true.pos.preds <- tree.preds[test.pca$Class == 4]
tree.sensitivity <- sum(tree.true.pos == tree.true.pos.preds) / length(tree.t
rue.pos)
# Calculate the specificity for the tree model.
tree.true.neg <- test.pca[test.pca$Class == 2, "Class"]
```

```
tree.true.neg.preds <- tree.preds[test.pca$Class == 2]
tree.specificity <- sum(tree.true.neg == tree.true.neg.preds) / length(tree.t
rue.neg)

# Calculate the accuracy for the knn model.
knn.accuracy <- sum(knn.preds == test.pca$Class) / length(knn.preds)
# Calculate the APER for the knn model.
knn.aper <- sum(knn.preds != test.pca$Class) / length(knn.preds)
# Calculate the sensistivity for the knn model.
knn.true.pos <- test.pca[test.pca$Class == 4, "Class"]
knn.true.pos.preds <- knn.preds[test.pca$Class == 4]
knn.sensitivity <- sum(knn.true.pos == knn.true.pos.preds) / length(knn.true.
pos)
# Calculate the specificity for the knn model.
knn.true.neg <- test.pca[test.pca$Class == 2, "Class"]
knn.true.neg.preds <- knn.preds[test.pca$Class == 2]
knn.specificity <- sum(knn.true.neg == knn.true.neg.preds) / length(knn.true.
neg)
```

The Classification Tree and KNN model trained on the first two principal components scores had a test set accuracy of 95.2%. Conversely, they have an apparent error rate of 4.8%. The sensitivity, or true positive rate, is 98.5% and the specificity, or true negative rate, is 93.8%. These models have a higher sensitivity than specificity; they are more likely to correctly classify a malignant mass than a benign one. These models are more prone to false positives than false negatives.

```
test.performance <- matrix(c(tree.accuracy, tree.aper, tree.sensitivity, tree
.specificity,
                             knn.accuracy, knn.aper, knn.sensitivity, knn.spec
ificity),
                           nrow=2, byrow=TRUE)
rownames(test.performance) <- c(paste("Classification Tree (cp = ", best.cp,
")", sep=""),
                                paste("KNN (k = ", best.k, ")", sep=""))
colnames(test.performance) <- c("Accuracy", "APER", "Sensitivity", "Specifici
ty")
test.performance <- as.data.frame(test.performance)

knitr::kable(test.performance, align = "c",
             caption =  "Test Set Peformance", digits = 5)
```

Test Set Peformance

	Accuracy	APER	Sensitivity	Specificity
Classification Tree (cp = 0.5)	0.95238	0.04762	0.98485	0.9375
KNN (k = 3)	0.95238	0.04762	0.98485	0.9375

```
# Set the proportion of tree predictions equal to knn predictions.
proportion.equal <- sum(tree.preds == knn.preds) / length(tree.preds)
```

```
print(paste("Percent of Tree Model Predictions equal to K-NN Model Prediction
s: ",
            round(100*proportion.equal, 5), "%", sep=""))
```

```
## [1] "Percent of Tree Model Predictions equal to K-NN Model Predictions: 10
0%"
```

Both false positives and false negatives have very high costs when determining whether a breast mass is benign or malignant. A false positive (a benign mass classified as malignant) could result in an unnecessary treatment for cancer, which carries health and financial costs. A false negative (a malignant mass classified as benign) could delay a proper diagnosis and possibly result in death.

Conclusions