Untitled

John Clements

11/14/2020

Part 1: Data Loading, Cleaning, and Splitting

```
#Breast Cancer Detection Project
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------ tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts --------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
##### DATA CLEANING #####

# Read in data set.
setwd("C:/Users/17739/Documents/GitHub/BreastCancerDetection/Breast Cancer Git Hub/")
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')

#head(bc.df) # View first 6 rows of data.
#dim(bc.df) # Check the dimensions of the data.
#str(bc.df) # Bare Nuclei is only feature stored as chr instead of int.
#table(bc.df$Class) # Report the frequency for each class.

# Write dataframe to a csv file.
#write.csv(bc.df,"breast-cancer-wisconsin.csv", row.names = FALSE)

# Identify missing data.
#bc.df %>% filter(!`Bare Nuclei` %in% c("1","2","3","4","5","6","7","8","9","10"))

# Calculte 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)
```

```
## Warning in mean(as.numeric(`Bare Nuclei`), na.rm = TRUE): NAs introduced by
## coercion
```

```
## Warning in mean(as.numeric(`Bare Nuclei`), na.rm = TRUE): NAs introduced by
## coercion
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

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

```
## Warning in mean(as.numeric(`Bare Nuclei`), na.rm = TRUE): NAs introduced by
## coercion
```

```
## Warning in mean(as.numeric(`Bare Nuclei`), na.rm = TRUE): NAs introduced by
## coercion
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# Impute missing Bare Nuclei data with the mean for each Class mean.
#Bare Nuclei (class 2) with mean Bare Nuclei for class 2
#Bare Nuclei (class 4) with mean Bare Nuclei for class 4
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`))) #%>%
  #filter(`Bare Nuclei` == "?")

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

#str(bc.df)

#write revised dataframe to csv
#write.csv(bc.df,"breast-cancer-wisconsin_revised.csv", row.names = FALSE)

# 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")))
```

```r
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')

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

Table 1: 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	1.34	7.62

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

Table 2: 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	1.161	3.104

```r
##### 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)
```

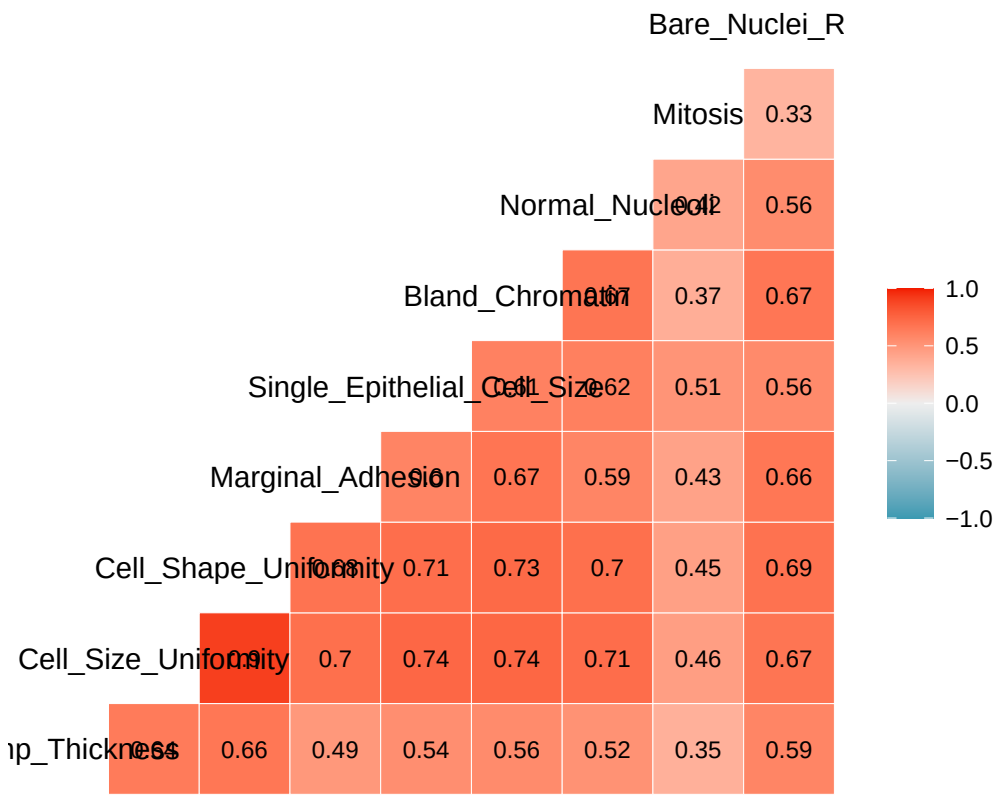
```r
# set the indicies of the dataset to be in the training data set.
train.indices <- sample(1:num.obs, num.train, replace=FALSE)

# Subset the data into training and testing data sets.
train.bc.df <- bc.df[train.indices,]
test.bc.df <- bc.df[-train.indices,]
```

Part 2: Principal Components Analysis

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

library(knitr)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
# 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 correlated (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                0.5929538             0.6744643
##                              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                      0.6877416         0.6625310
##                              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                          0.5642289       0.6657948
##                             Normal Nucleoli   Mitosis Bare Nuclei Revised
## Clump Thickness                   0.5235813 0.3544657           0.5929538
## Cell Size Uniformity              0.7106115 0.4570546           0.6744643
## Cell Shape Uniformity             0.6992053 0.4466957           0.6877416
## Marginal Adhesion                 0.5905170 0.4325498           0.6625310
## Single Epithelial Cell Size       0.6169662 0.5081960           0.5642289
## Bland Chromatin                   0.6729169 0.3678463           0.6657948
## Normal Nucleoli                   1.0000000 0.4185491           0.5554505
## Mitosis                           0.4185491 1.0000000           0.3340189
## Bare Nuclei Revised               0.5554505 0.3340189           1.0000000
```

```
# Check the sd for each variable.
round(apply(train.bc.df.X, 2, sd),3) # need to standardize variables
```

```
##             Clump Thickness        Cell Size Uniformity
##                       2.861                       3.030
##       Cell Shape Uniformity           Marginal Adhesion
##                       2.903                       2.920
## Single Epithelial Cell Size             Bland Chromatin
##                       2.158                       2.447
##             Normal Nucleoli                     Mitosis
##                       3.086                       1.615
##         Bare Nuclei Revised
##                       3.650
```

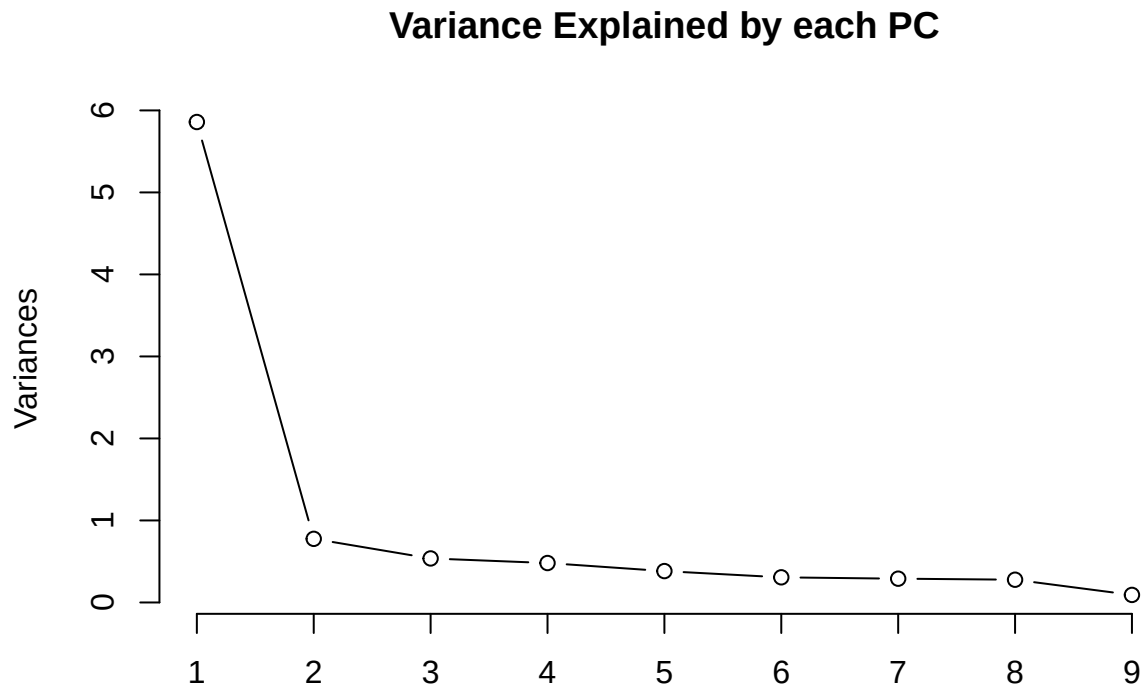
```
# Standardize the variables.
train.bc.df.X.std <- scale(train.bc.df.X, center=TRUE, scale=TRUE)
# Check if the data is Scaled Correctly.
apply(train.bc.df.X.std, 2,sd)
```

```
##             Clump Thickness        Cell Size Uniformity
##                           1                           1
##       Cell Shape Uniformity           Marginal Adhesion
##                           1                           1
## Single Epithelial Cell Size             Bland Chromatin
##                           1                           1
##             Normal Nucleoli                     Mitosis
##                           1                           1
##         Bare Nuclei Revised
##                           1
```

```
# Perform PCA.
data.pca <- prcomp(train.bc.df.X.std)
# Extract the importance of each component.
summary(data.pca)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4    PC5     PC6     PC7
## Standard deviation     2.4204 0.88077 0.73263 0.69347 0.6185 0.55355 0.53849
## Proportion of Variance 0.6509 0.08619 0.05964 0.05343 0.0425 0.03405 0.03222
## Cumulative Proportion  0.6509 0.73712 0.79676 0.85019 0.8927 0.92674 0.95896
##                            PC8     PC9
## Standard deviation     0.52661 0.30335
## Proportion of Variance 0.03081 0.01022
## Cumulative Proportion  0.98978 1.00000
```

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

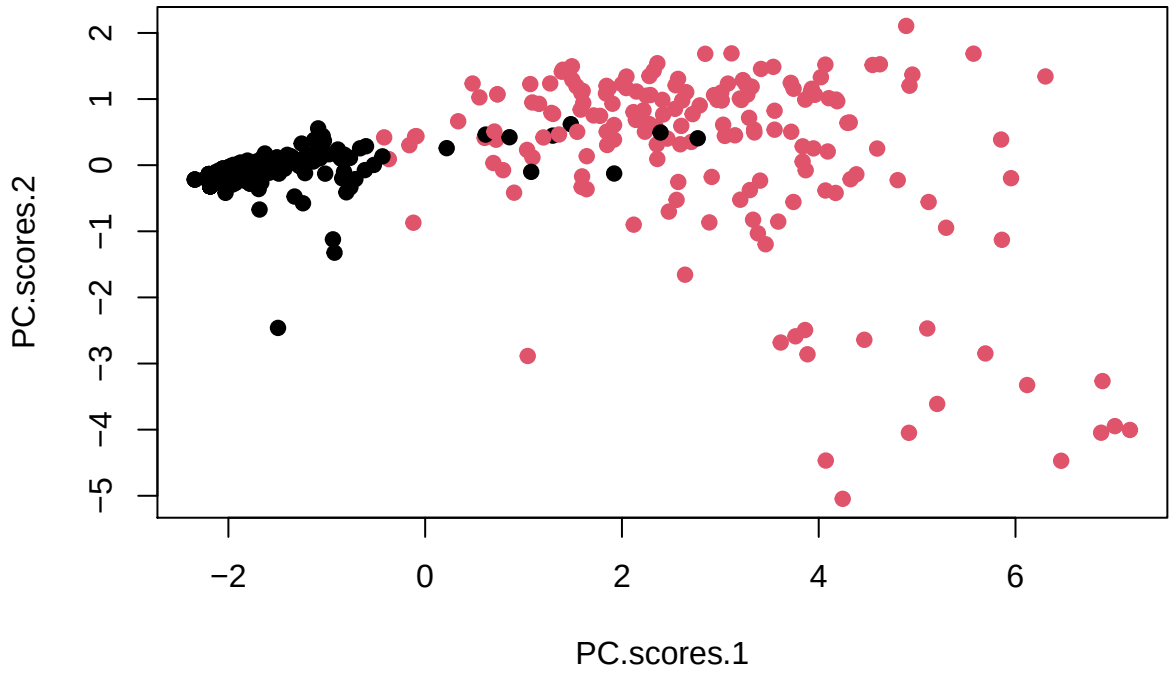



```
# The elbow in the graph is at PC2.
#PC 1 and 2 explain 74% of the variability

# Display the loadings for PC1 and PC2 for review.
round(data.pca$rotation[, 1:2], 3)
```

```
##                               PC1    PC2
## Clump Thickness             0.306  0.164
## Cell Size Uniformity        0.380  0.058
## Cell Shape Uniformity       0.377  0.087
## Marginal Adhesion           0.334  0.039
## Single Epithelial Cell Size 0.337 -0.217
```

```
## Bland Chromatin              0.347  0.201
## Normal Nucleoli              0.332 -0.003
## Mitosis                      0.235 -0.889
## Bare Nuclei Revised          0.330  0.288
```

```
# 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)
```



Part 4: Classification

```
# 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)
```

```r
# 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
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 4.0.3

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

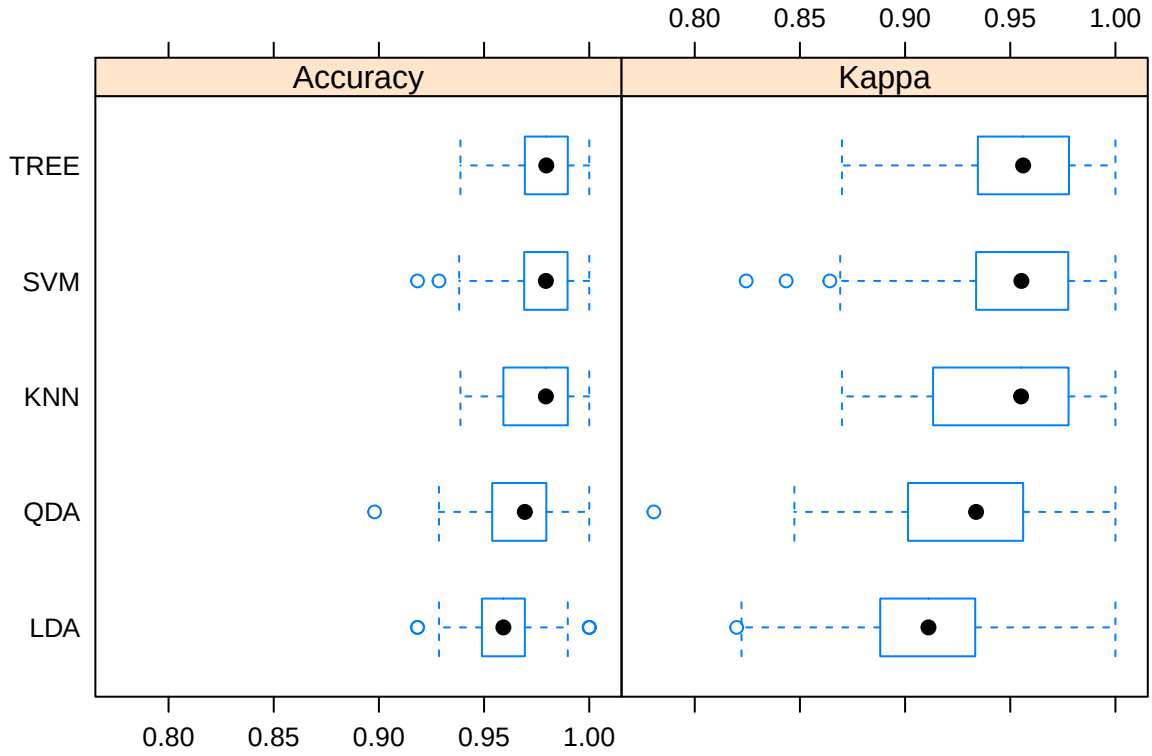
```r
# We will be using 5-fold cross-validation, repeating the process 20 times.
TrControl <- trainControl(method = "repeatedcv",
                          number = 5,
                          repeats = 20)

# Fit a KNN classifier by selecting the model with the best number of neighbors
# 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.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 <- train(Class ~ ., data = , train.pca,
```

```
                  method = "lda",
                  trControl = TrControl)

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

# Fit a tree classifier by selecting the model with the best complexity parameter (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.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 <- 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 <- resamples(list(KNN = knn.model, LDA = lda.model, QDA = qda.model,
                         TREE = tree.model, SVM = svm.model))

# Display the summary of the model performances.
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: KNN, LDA, QDA, TREE, SVM
## Number of resamples: 100
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## KNN  0.9387755 0.9591837 0.9793814 0.9751546 0.9897959    1    0
## LDA  0.9183673 0.9489796 0.9591837 0.9609436 0.9693878    1    0
## QDA  0.8979592 0.9563171 0.9693878 0.9666653 0.9795918    1    0
## TREE 0.9387755 0.9693878 0.9795918 0.9780255 0.9897959    1    0
## SVM  0.9183673 0.9690722 0.9793814 0.9738186 0.9896907    1    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## KNN  0.8699690 0.9133127 0.9551417 0.9464722 0.9776738    1    0
## LDA  0.8199357 0.8881789 0.9111111 0.9146458 0.9331190    1    0
## QDA  0.7805643 0.9062384 0.9337539 0.9282077 0.9561129    1    0
```

```
## TREE 0.8699690 0.9345794 0.9561129 0.9529358 0.9779180    1    0
## SVM  0.8244514 0.9337539 0.9552995 0.9437805 0.9775413    1    0
```

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




```
library(klaR)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

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

```
# Get predictions for the test set from the tree model.
tree.preds <- predict(tree.model, test.pca)
# Make a confusion matrix for the tree model.
tree.emat <- errormatrix(test.pca$Class, tree.preds,
                         relative = TRUE)
# Display the confusion matrix for the tree model.
round(tree.emat, 3)
```

```
##        predicted
## true        2     4 -SUM-
##   2     0.944 0.056 0.056
##   4     0.015 0.985 0.015
##   -SUM- 0.111 0.889 0.043
```

```r
# Get predictions for the test set from the knn model.
knn.preds <- predict(knn.model, test.pca)
# Make a confusion matrix for the knn model.
knn.emat <- errormatrix(test.pca$Class, knn.preds,
                        relative = TRUE)
# Display the confusion matrix for the knn model.
round(knn.emat, 3)
```

```
##        predicted
## true        2     4 -SUM-
##   2     0.938 0.062 0.062
##   4     0.015 0.985 0.015
##   -SUM- 0.100 0.900 0.048
```