Iris Plant Classification Using Machine Learning

Chris Meade

11/6/2016

1. Introduction

About the Data Set

The iris data set contains 150 observations of 5 variables. The species of the plant, `Species` is the response variable to be predicted from the other four variables: `Sepal.Length`, `Sepal.Width`, `Petal.Length`, and `Petal.Width`.

Importing the Data

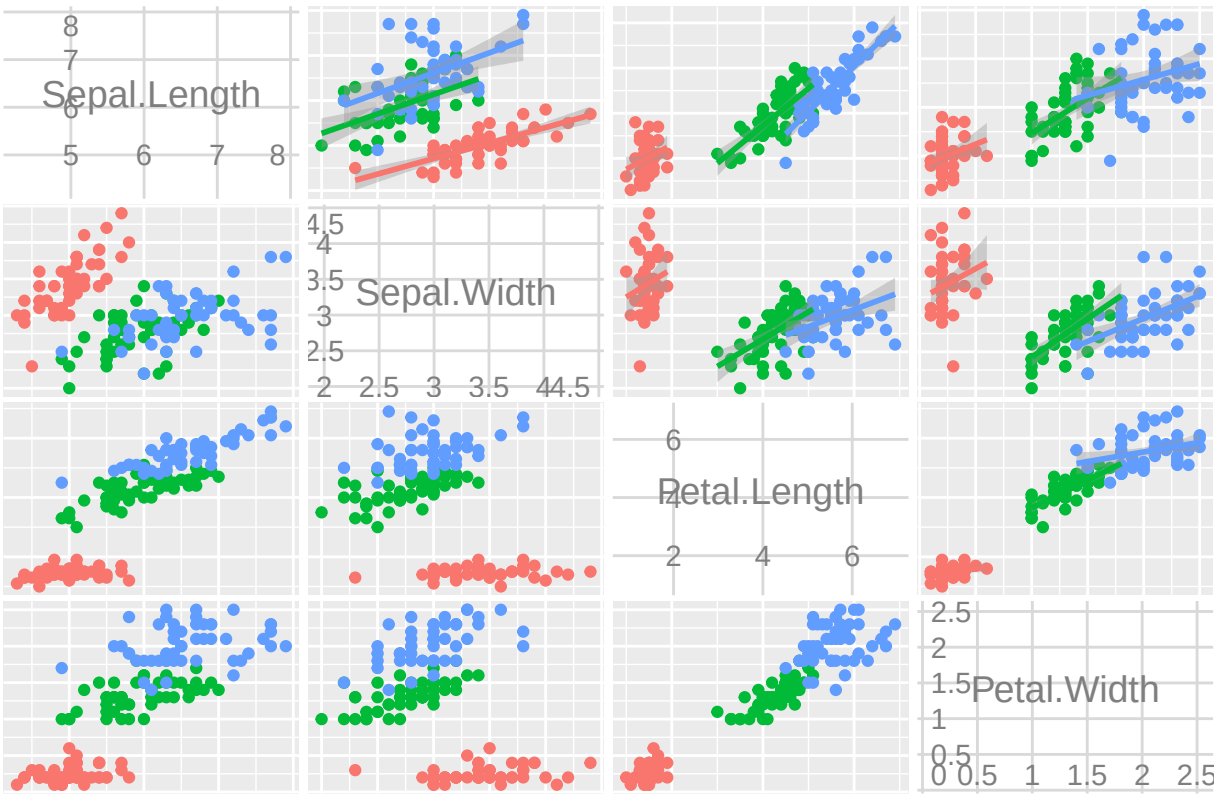
```
data(iris)
iris = as.data.frame(iris)
```

Exploratory Data Analysis

We first begin by examing how the variables interact with one another

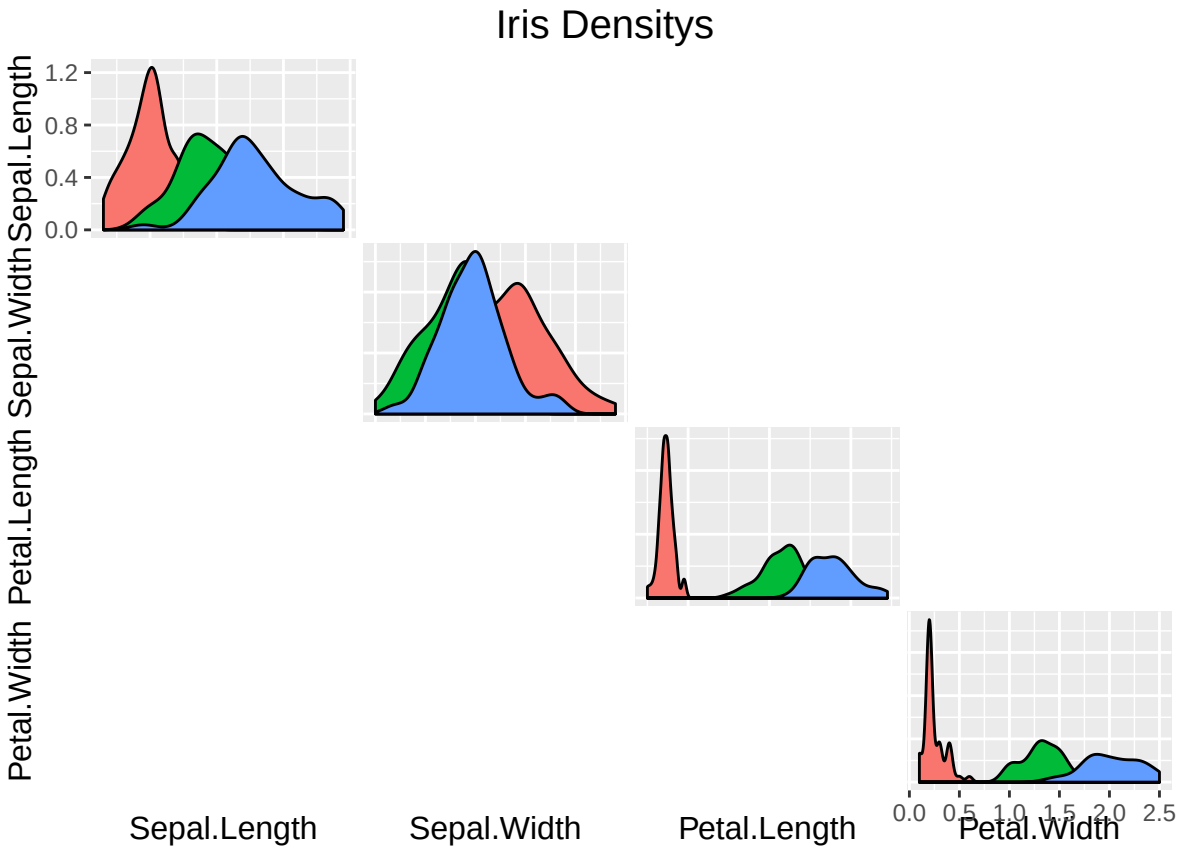
```
scatterPlot <- ggpairs(data=iris,
                columns=1:4,
                upper = list(continuous = wrap("smooth")),
                lower = list(continuous = wrap("points"), combo = wrap("dot")),
                aes(color = Species),
                axisLabels = "internal",
                title = "Iris Scatterplot Matrix")

scatterPlot
```




```
scatterPlot2 <- ggpairs(data=iris,
                        columns=1:4,
                        upper = "blank",
                        lower = "blank",
                        aes(color = Species),
                        title = "Iris Densitys")

scatterPlot2
```



We make the following observations that guide our analysis:

1. There appears to be clear separation of the observations by species when the predicator variables are compared. These makes this data set a prime candidate for machine learning prediction.
2. `Sepal.Length`, `Petal.Length`, and `Petal.Width` are highly correlated. `Sepal.Width` is not.

Split the Data into a Training Set and Test Set

```
# Separate predictor variables from Species
X.iris = iris[, c('Sepal.Length', 'Sepal.Width', 'Petal.Length', 'Petal.Width')]
Y.iris = iris[, 'Species']

# Reserve 30% of the data for a test set
train.percent <- 0.70
train.number <- train.percent * nrow(iris)
train.indices <- sample.int(nrow(iris), train.number)

# Extract the training set using our train indices
X.train <- X.iris[train.indices,]
Y.train <- Y.iris[train.indices]
train.data <- iris[train.indices,]

# Get the test set from the rest
X.test <- X.iris[-c(train.indices),]
```

```
Y.test <- Y.iris[-c(train.indices)]
test.data <- iris[-c(train.indices),]

# Set seed
set.seed(131)
```

Model Construction

K-Nearest Neightbors

Description of knn

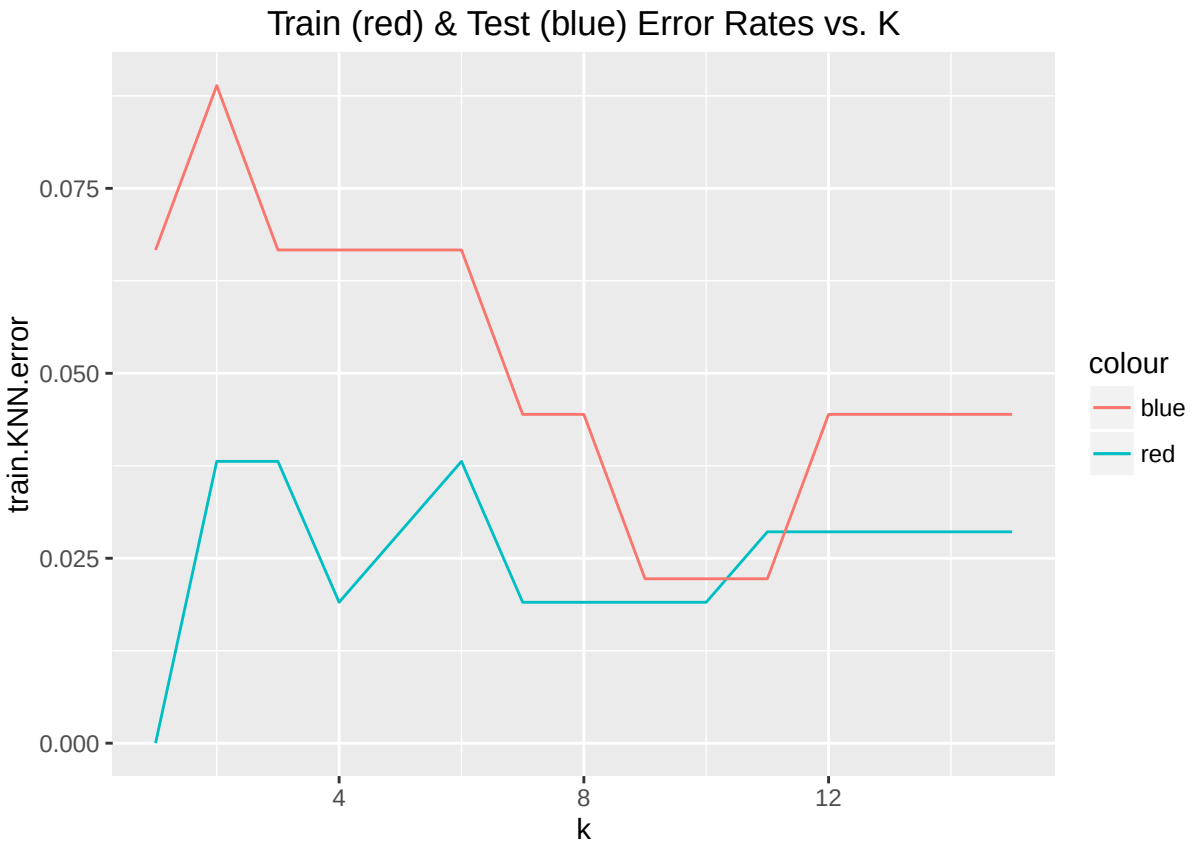
```
# Build he Model for k=1:15

#Calculate Training Error Rates

train.KNN.error = rep(0,15)
k = 1:15
for(i in k ){
  model <- knn(X.train,
               X.train,
               Y.train,
               k = i)
  train.KNN.error[i] <- mean(model != Y.train)
}

#Calculate Test Error Rates
test.KNN.error = rep(0,15)
k = 1:15
for(i in k ){
  model <- knn(X.train,
               X.test,
               Y.train,
               k = i)
  test.KNN.error[i] <- mean(model != Y.test)
}
error.df <- data.frame(k, train.KNN.error, test.KNN.error)


ggplot(error.df) +
  geom_path(aes(k, train.KNN.error, color = 'red')) +
  geom_path(aes(k, test.KNN.error, color = 'blue'))+
  ggtitle("Train (red) & Test (blue) Error Rates vs. K")
```



Based on the graph on error rates, `k = 4` appears to provide a good compromise between not overfitting the data and providing accurate predictions of the test set. So with our KNN model, we have the following performance:

Model Analysis

```
# Error Rates
error.df[4,]
```

```
##   k train.KNN.error test.KNN.error
## 4 4      0.01904762     0.06666667
```

```
#ADD More shit
```

Linear Discriminant Analysis

Describe this shit

```
# Construct the Model
train.LDA <- lda(Species ~ ., data = train.data)

# Predict the Species class of the training set
predict.train.LDA <- predict(train.LDA, train.data)

# Calculate Training Test Rate
train.LDA.error<- mean(predict.train.LDA$class != train.data$Species)
train.LDA.error
```

```
## [1] 0.00952381
```

```
# Use Leave-one-out cross-validation to calculate test error rate
test.LDA <- lda(Species ~ ., data = iris, CV = T)

test.LDA.error <- mean(test.LDA$class != iris$Species)

test.LDA.error
```

```
## [1] 0.02
```

LDA yields a test error rate of training error rate of 0.01904762 and test error rate of 0.02.

QDA

Talk about QDA

```
train.QDA <- qda(Species ~ ., data = train.data)
predict.train.QDA <- predict(train.QDA, train.data)
train.QDA.error <- mean(predict.train.QDA$class != train.data$Species)
train.QDA.error
```

```
## [1] 0.00952381
```

```
test.QDA <- lda(Species ~ ., data = iris, CV = T)
test.QDA.error <- mean(test.QDA$class != iris$Species)
test.QDA.error
```

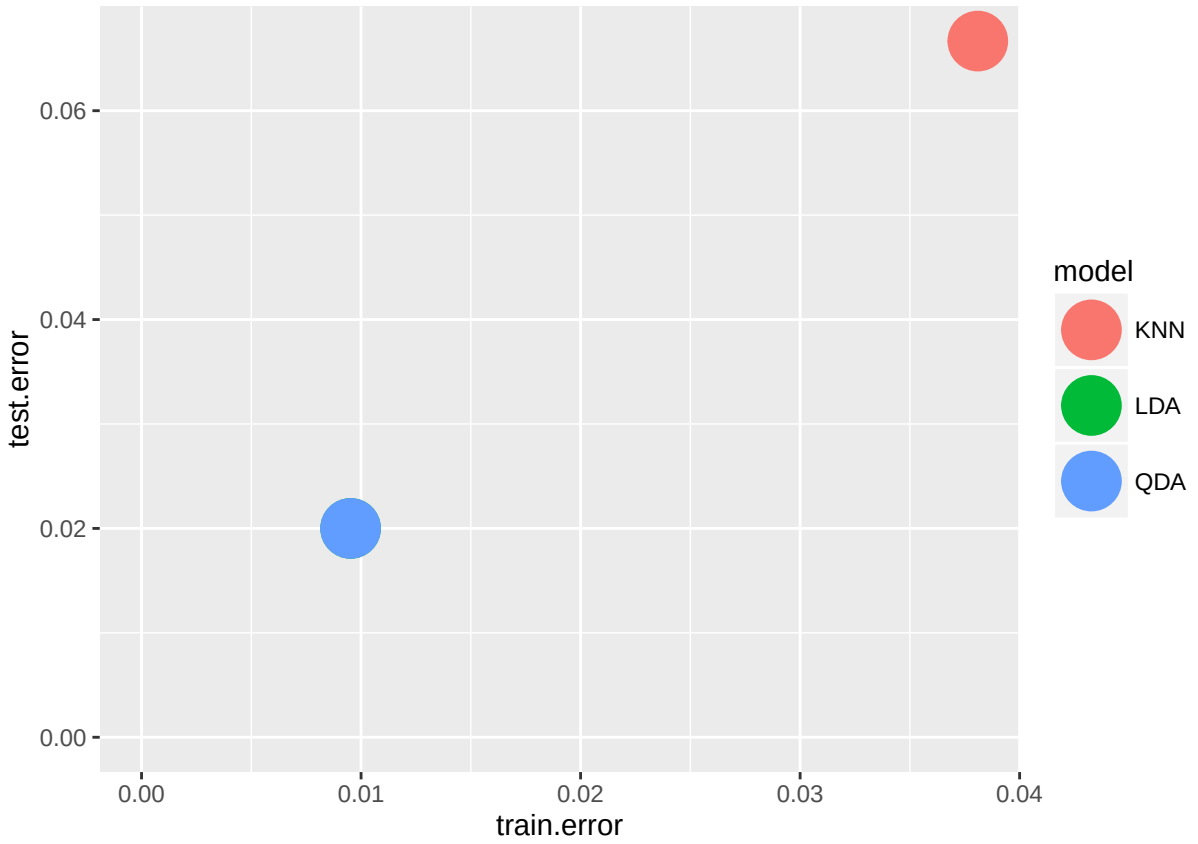
```
## [1] 0.02
```

```
#Construct a data table or training and test error rates

KNN.df <- error.df[3,2:3]
names(KNN.df) <- c("train.error", "test.error")
LDA.df <- data.frame(train.error = train.LDA.error, test.error = test.LDA.error)
QDA.df <- data.frame(train.error = train.QDA.error, test.error =test.QDA.error)

model.errors <- rbind(KNN.df, LDA.df, QDA.df)
model.errors <- data.frame(model = c("KNN", "LDA", "QDA"), model.errors)

error.plot <- ggplot(model.errors, aes(x = train.error, y = test.error, color = model))
error.plot <- error.plot + geom_point(size = 10) + expand_limits(x = 0, y = 0)
error.plot
```



Based on error rates, we see that QDA best model. Minimized training error, tied for best test error with LDA. Possible LDA may be better with some samples. Use bootstrap to confirm our hypothesis.

Bootstrap Method

```
#Create empty vectors to append error rate in each iteration of the for loop
knnErrors = vector()
ldaErrors = vector()
qdaErrors = vector()

set.seed(131)
#Run each model 100 times
for (i in 1:100){
  #Create new dataset of same size with random sample of iris data
  sampleIndex <- sample(1:nrow(iris), size = nrow(iris), replace = TRUE)
  sampleData <- iris[sampleIndex, ]

  # Fit the three models to the new sample data
  KNNfit <- knn.cv(train = sampleData[, 1:4], cl = sampleData[, 5], k = 14)
  LDAfit <- lda(Species ~ ., data = sampleData, CV = T)
  QDAfit <- qda(Species ~ ., data = sampleData, CV = T)

  # Compute error rates for each iteration
  knnError <- mean(KNNfit != sampleData$Species)
  ldaError <- mean(LDAfit$class != sampleData$Species)
  qdaError <- mean(QDAfit$class != sampleData$Species)
```

```
  # Save error rates for each iteration
  knnErrors[i] = knnError
  ldaErrors[i] = ldaError
  qdaErrors[i] = qdaError
}

# Aggregate results
knnMean <- mean(knnErrors)
knnVar <- var(knnErrors)

ldaMean <- mean(ldaErrors)
ldaVar <- var(ldaErrors)

qdaMean <- mean(qdaErrors)
qdaVar <- var(qdaErrors)

meanKNN <- paste("Mean KNN Error Rate =", knnMean)
varKNN <- paste("Variance of KNN Error Rate =", knnVar)

meanLDA <- paste("Mean LDA Error Rate =", ldaMean)
varLDA <- paste("Variance of LDA Error Rate =", ldaVar)

meanQDA <- paste("Mean QDA Error Rate =", qdaMean)
varQDA <- paste("Variance of QDA Error Rate =", qdaVar)

cat(paste(meanKNN,varKNN,meanLDA,varLDA,meanQDA,varQDA, sep = '\n'))
```

```
## Mean KNN Error Rate = 0.0369333333333333
## Variance of KNN Error Rate = 0.000312835016835017
## Mean LDA Error Rate = 0.0214666666666667
## Variance of LDA Error Rate = 0.000136996632996633
## Mean QDA Error Rate = 0.0194
## Variance of QDA Error Rate = 0.000121297418630752
```