Predictive Model to Classify Flowers

By: Jeremy Kight

Date: 11/24/2019

Contents

Abstract:

The objective of this project is to accurately predict the species of a flower based on the measurements of a flowers sepals and petals. The iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. Various models were tested with a 10-fold cross validation on a sample of the data to determine which type of model would give the best results. It was determined that a linear model would work best. The linear model created and used in this report accurately predicts the class of flower 98.3% of the time.

Introduction:

This is one of the most well know data sets among beginners in R programming. The aim of this project is to classify iris flowers among three species (setosa, versicolor or virginica) from measurements of length and width of sepals and petals. The following code is designed to create a model which makes good classifications for new flowers or, in other words, one which exhibits good generalization. This code is only meant to scratch the surface in terms of machine learning but it shows the usefulness such a code could provide for those who study plants/flowers.

Procedure:

Loading In The Required Libraries

```r
library(tidyverse)
library(caret)
library(ggplot2)
```

Loading In and Viewing The Data

The Iris Flower data set is include in R or it can be downloaded from UCI Machine Learning Repository. In this case, it was downloaded from UCI Machine Learning Repository and saved locally. Running the code below will then upload the data into R.

```
dataset <- read_csv("iris_flowers.csv")
```

Now, that the data set is loaded into R, it can be viewed View(). There are many options to get an idea of what the data set contains. In this case, View() is not a bad option because the data set is relatively small.

```
view(dataset)
```

While looking at the table, one could decipher what the class types are in the data set. However, the sapply() function could do it quicker and more accurately.

```
sapply(dataset, class)
```

```
## sepal_length  sepal_width petal_length  petal_width        class
##    "numeric"    "numeric"    "numeric"    "numeric"  "character"
```

The Iris Flower data set contains four numeric classes and one character class. Now, using the table() function, the amount of factors can be determined.

```
table(dataset$class)
```

```
##
##     iris_setosa iris_versicolor  iris_virginica
##              50              50              50
```

There appears to be three factors in the class column. However, the class column contains character values. So the class column needs to be converted into a factor.

```
dataset$class <- as.factor(dataset$class)
head(dataset)
```

```
## # A tibble: 6 x 5
##   sepal_length sepal_width petal_length petal_width class
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 iris_setosa
## 2          4.9         3            1.4         0.2 iris_setosa
## 3          4.7         3.2          1.3         0.2 iris_setosa
## 4          4.6         3.1          1.5         0.2 iris_setosa
## 5          5           3.6          1.4         0.2 iris_setosa
## 6          5.4         3.9          1.7         0.4 iris_setosa
```

Splitting Data For Training and Testing

The purpose of splitting the data is to determine whether or not the model created can accurately predict outcomes based on certain parameters. If a model is exposed to all the data in the data set, then the model is no longer predicting… it is representing. The outcome should always be correct (or very close) because the program already knows which outputs correspond to each input. Thus, if new inputs were introduced to the machine learning algorithm, then the model could produce results that could be incorrect. This would have huge implications for businesses if they are expecting an algorithm to help with business decisions but the predictive model is actually giving the business detrimental advice.

First, to ensure the results are reproducable when using a random number generator, a seed needs to be set.

```
set.seed(3142)
```

The function createDataPartition can be used to create balanced splits of the data. If the y argument to this function is a factor, the random sampling occurs within each class and should preserve the overall class distribution of the data.

```
test_data <- createDataPartition(dataset$class, times = 1, p = 0.80, list = FALSE)
```

Next, split the data using a 80-20 split for training and validation. Then confirm there is an equal split. Lastly, summarize the split data set.

```
validation <- dataset[-test_data,] #Now select 20% of the data for validation.
dataset <- dataset[test_data,] #Now select 80% of the data for testing/training.
percentage <- prop.table(table(dataset$class)) * 100
print(percentage)
```

```
##
##     iris_setosa iris_versicolor  iris_virginica
##        33.33333        33.33333        33.33333
```

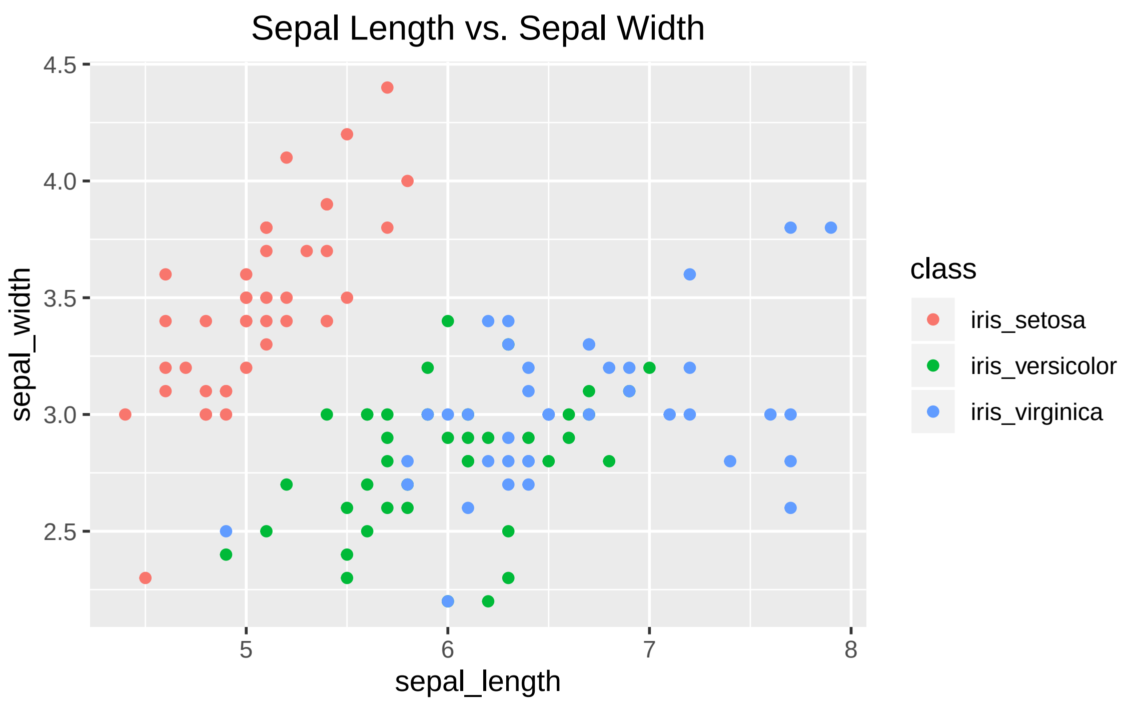
```
summary(dataset)
```

```
##   sepal_length    sepal_width    petal_length    petal_width
##  Min.   :4.400   Min.   :2.20   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.200   1st Qu.:2.80   1st Qu.:1.500   1st Qu.:0.300
##  Median :5.850   Median :3.00   Median :4.400   Median :1.300
##  Mean   :5.894   Mean   :3.09   Mean   :3.785   Mean   :1.208
##  3rd Qu.:6.400   3rd Qu.:3.40   3rd Qu.:5.125   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.40   Max.   :6.900   Max.   :2.500
##              class
##  iris_setosa    :40
##  iris_versicolor:40
##  iris_virginica :40
##
##
##
```

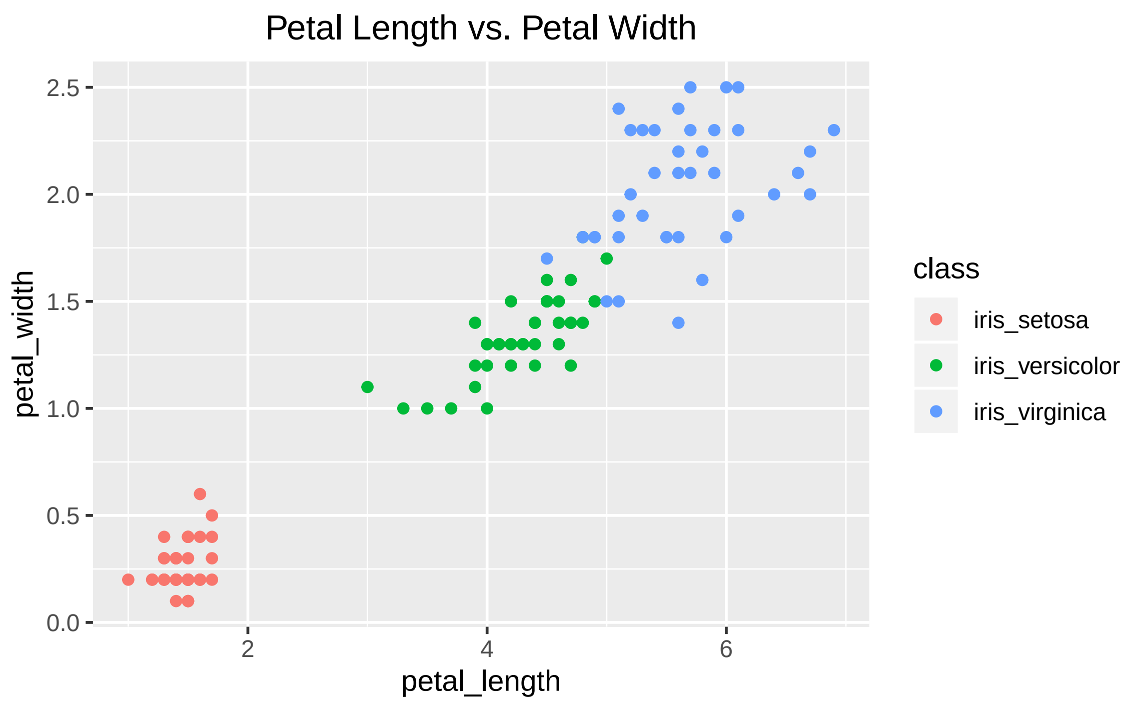
Visualizing The Data

The purpose of visualizing the data is to get an idea of whether the variables in the data set have a correlation. There are numerous methods for achieving this. That being said, the following code emphasizes ggplot( ), even though there could be more effective ways.

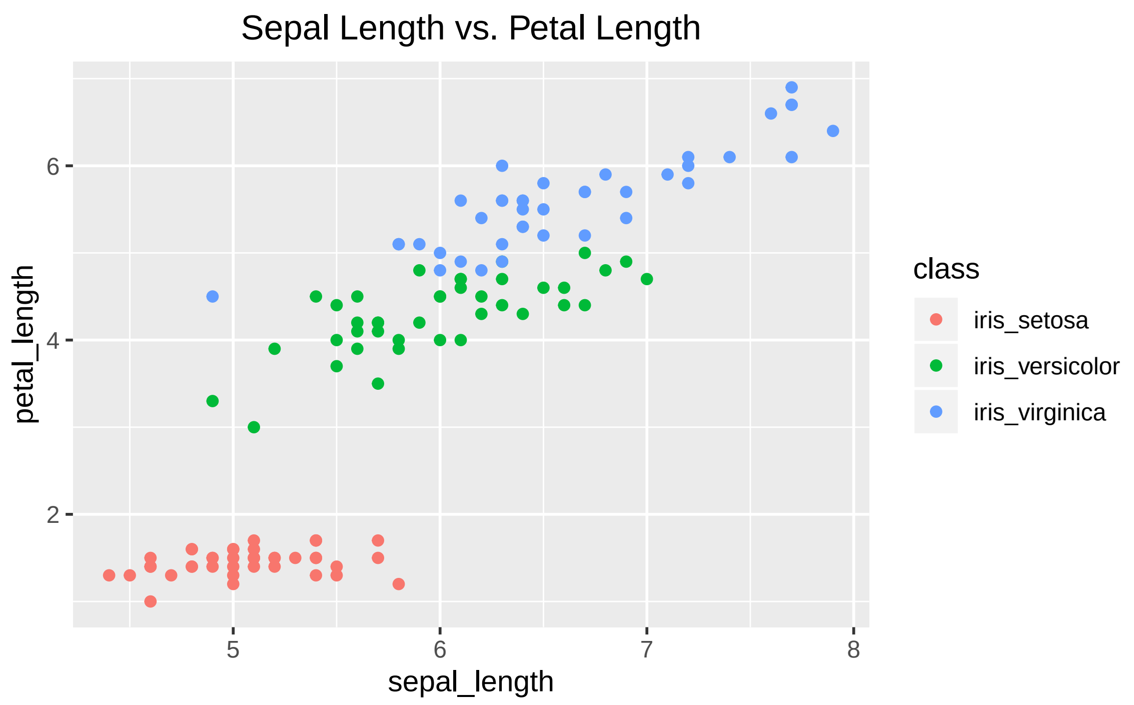
```r
ggplot(data = dataset) +
  geom_point(mapping = aes(x = sepal_length, y = sepal_width, color = class)) +
  ggtitle("Sepal Length vs. Sepal Width") +
  theme(plot.title = element_text(hjust = 0.5))
```



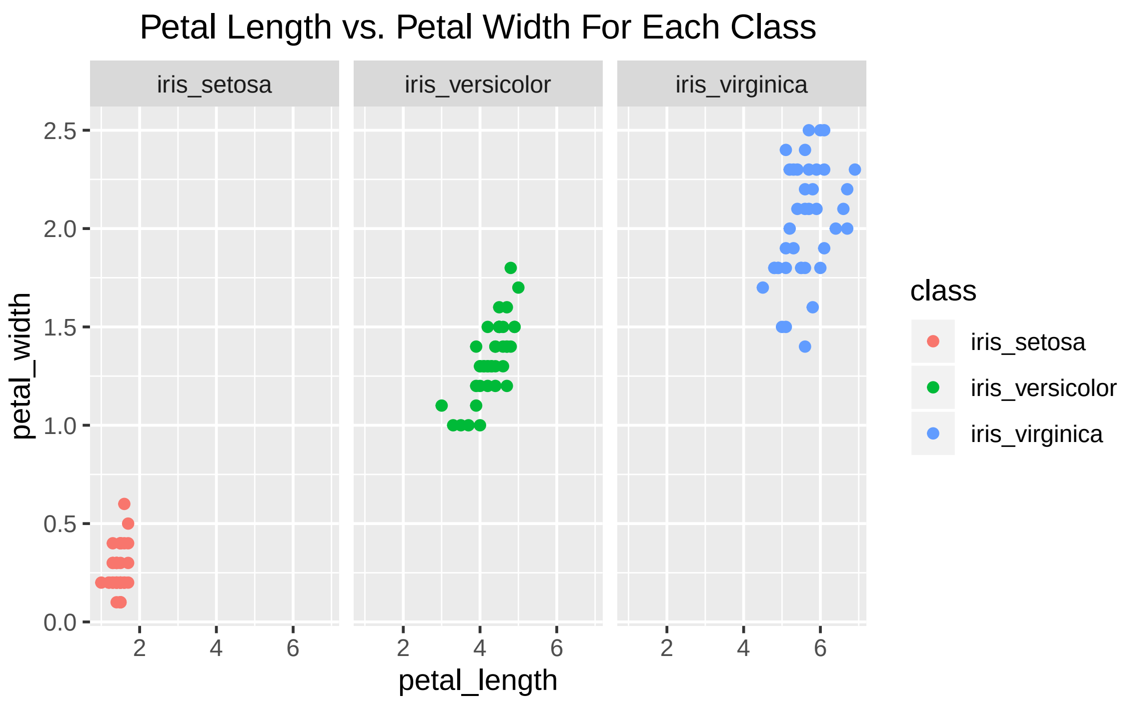
```r
ggplot(data = dataset) +
  geom_point(mapping = aes(x = petal_length, y = petal_width, color = class)) +
  ggtitle("Petal Length vs. Petal Width") +
  theme(plot.title = element_text(hjust = 0.5))
```



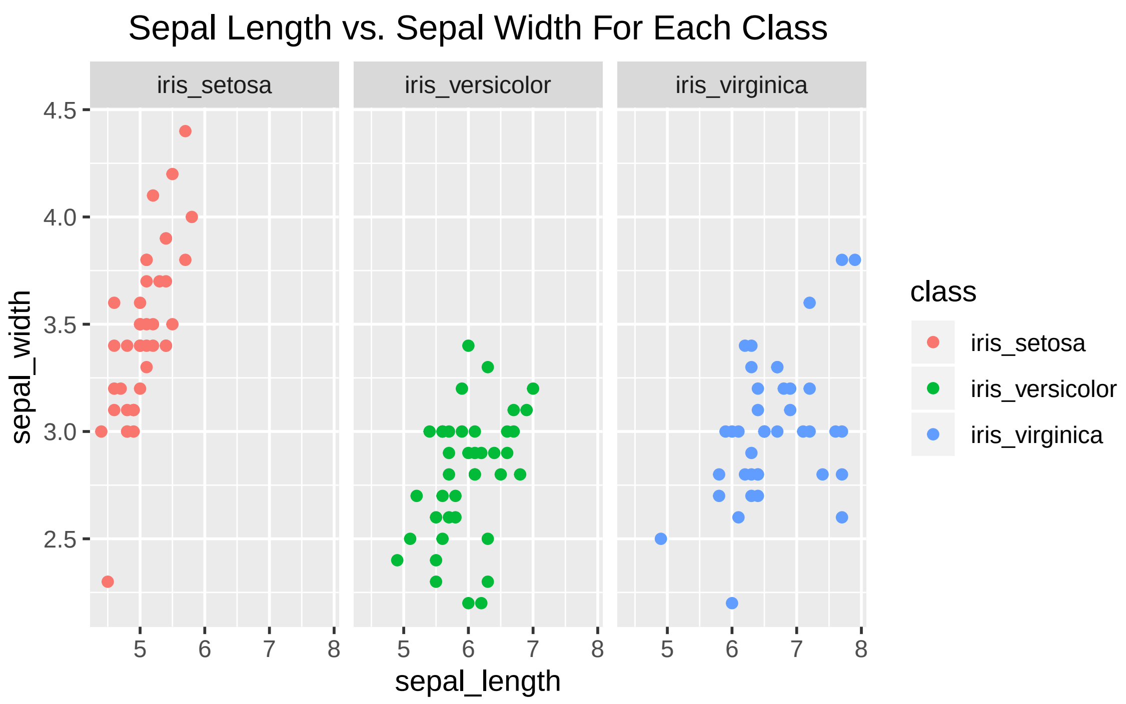
```
ggplot(data = dataset) +
  geom_point(mapping = aes(x = sepal_length, y = petal_length, color = class)) +
  ggtitle("Sepal Length vs. Petal Length") +
  theme(plot.title = element_text(hjust = 0.5))
```



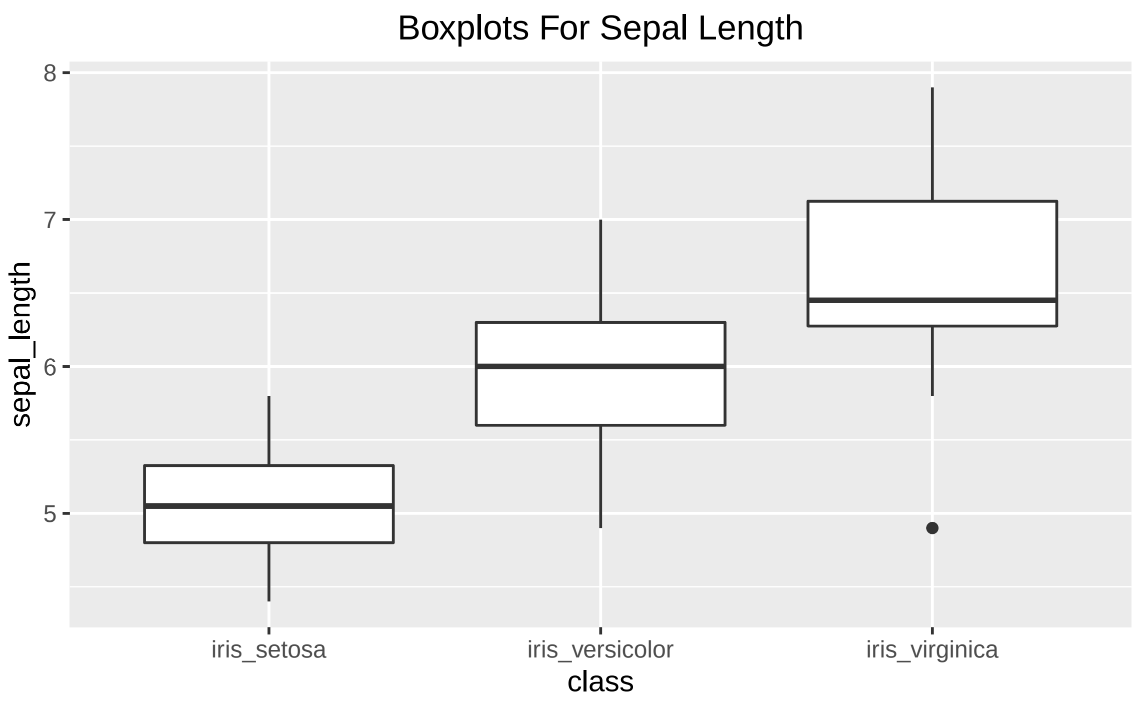
```r
ggplot(data = dataset) +
  geom_point(mapping = aes(x = petal_length, y = petal_width, color = class)) +
  facet_grid(~class)+
  ggtitle("Petal Length vs. Petal Width For Each Class")+
  theme(plot.title = element_text(hjust = 0.5))
```



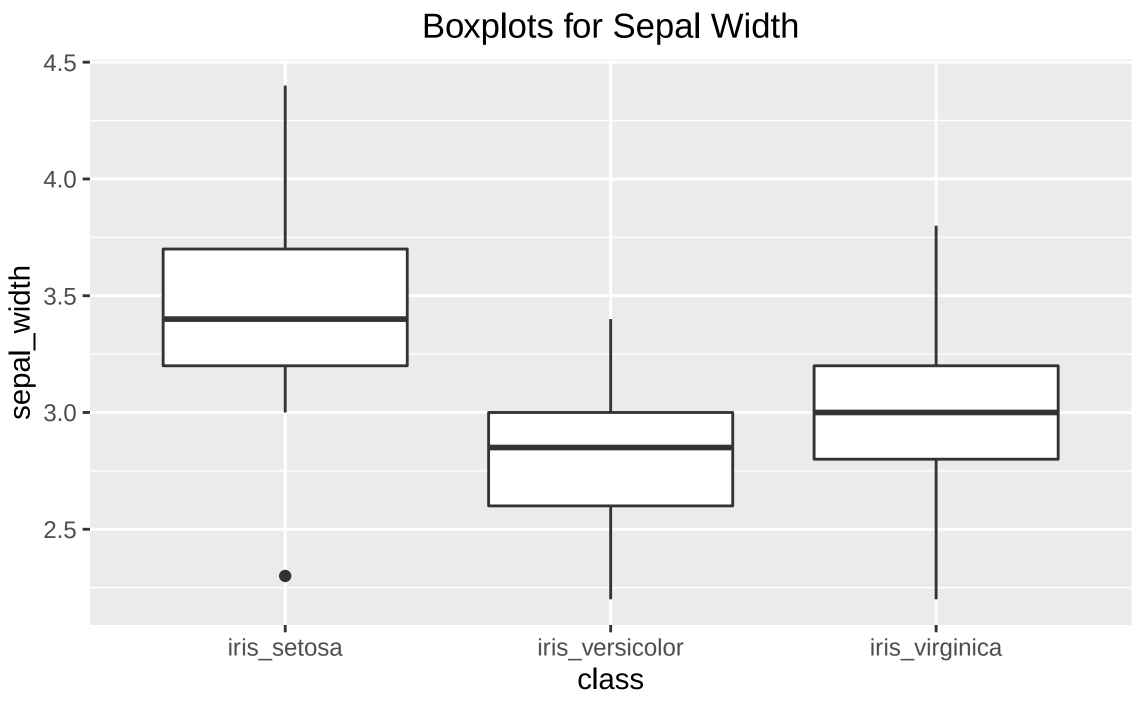
```r
ggplot(data = dataset) +
  geom_point(mapping = aes(x = sepal_length, y = sepal_width, color = class)) +
  facet_grid(~class)+
  ggtitle("Sepal Length vs. Sepal Width For Each Class")+
  theme(plot.title = element_text(hjust = 0.5))
```



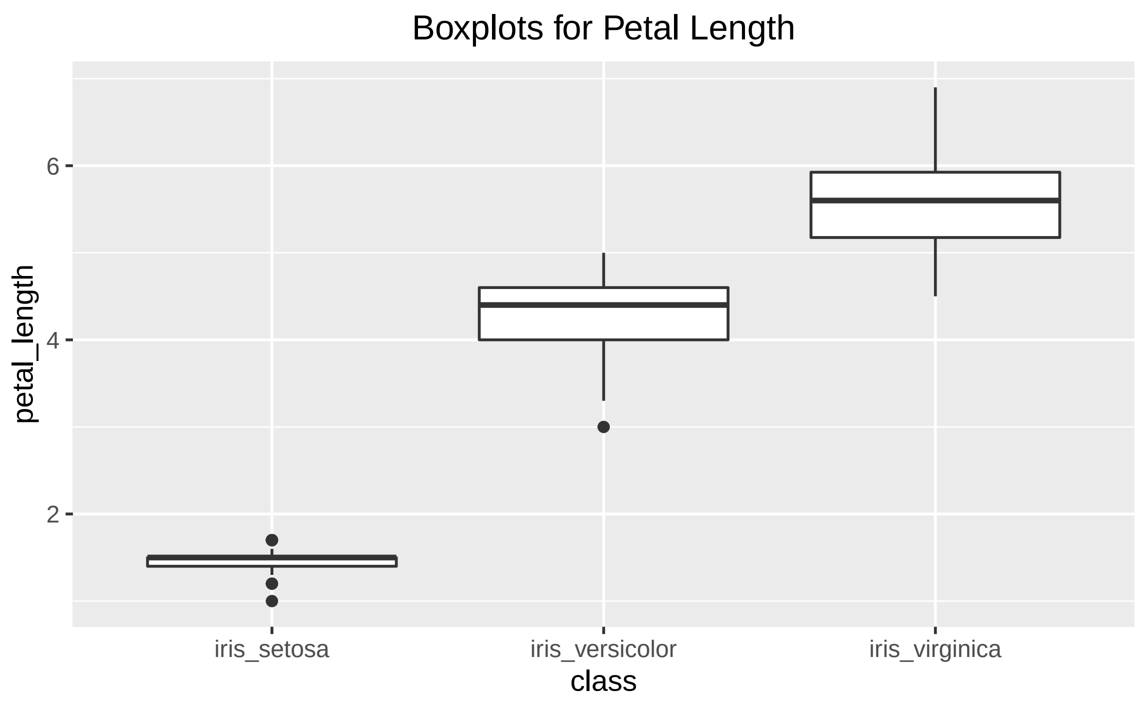
```r
ggplot(data = dataset, mapping = aes(x = class, y = sepal_length)) +
  geom_boxplot()+
  ggtitle("Boxplots For Sepal Length")+
  theme(plot.title = element_text(hjust = 0.5))
```



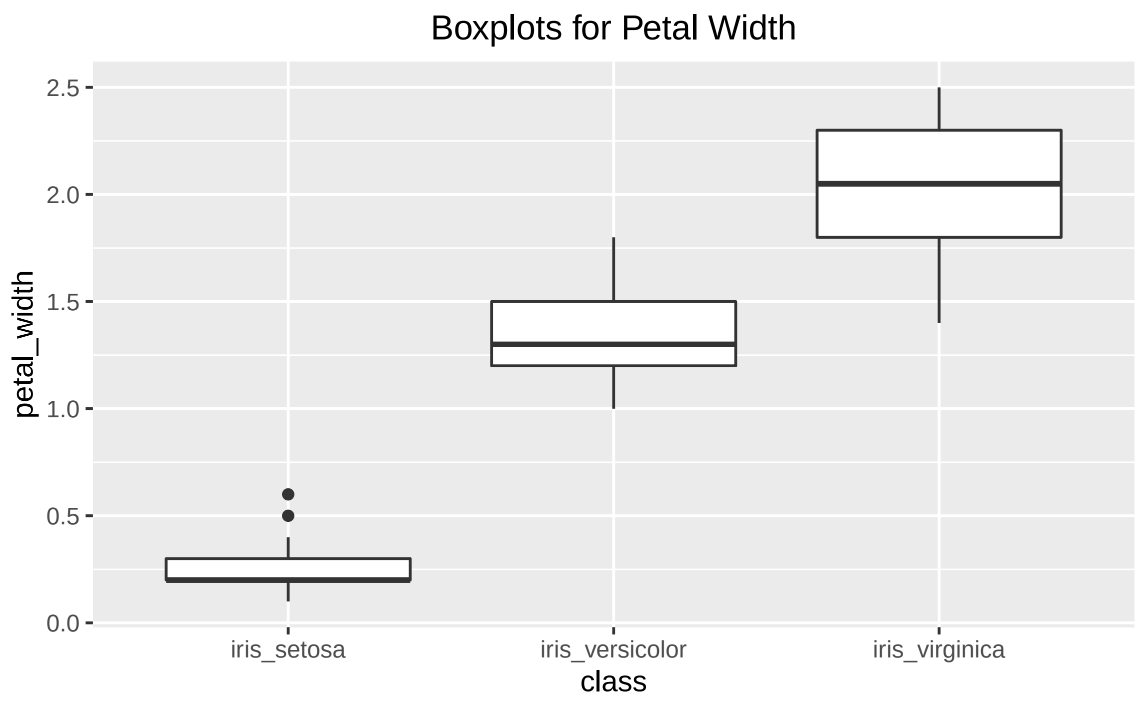
```r
ggplot(data = dataset, mapping = aes(x = class, y = sepal_width)) +
  geom_boxplot() +
  ggtitle("Boxplots for Sepal Width")+
  theme(plot.title = element_text(hjust = 0.5))
```



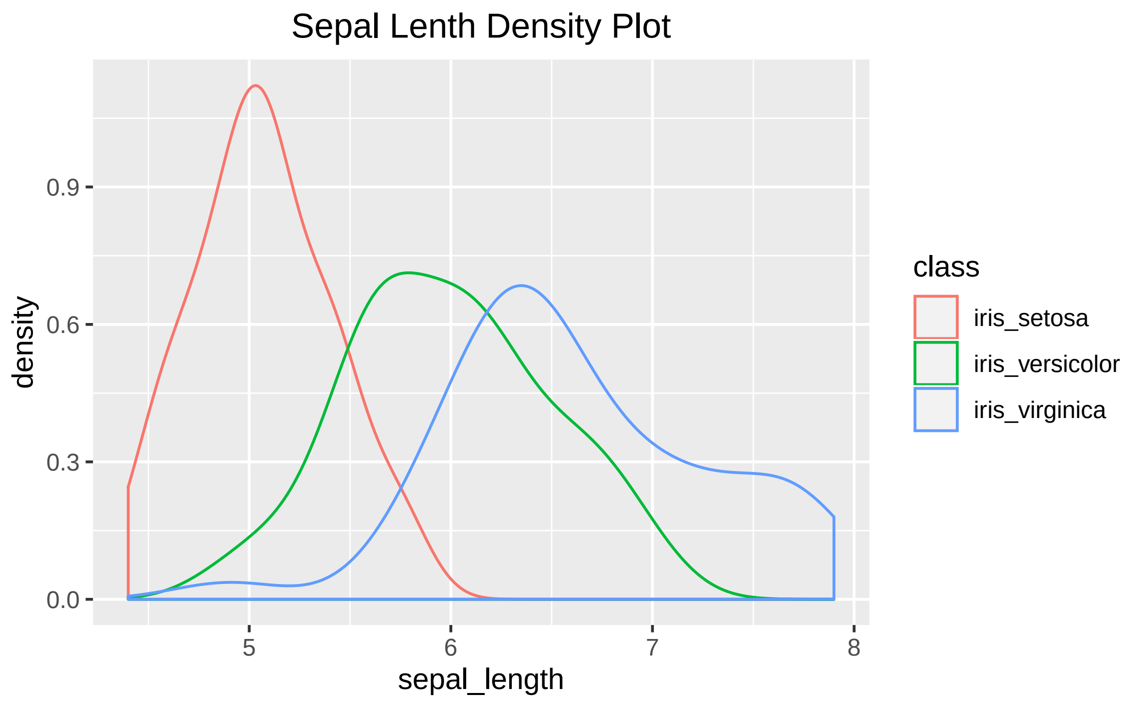
```r
ggplot(data = dataset, mapping = aes(x = class, y = petal_length)) +
  geom_boxplot() +
  ggtitle("Boxplots for Petal Length")+
  theme(plot.title = element_text(hjust = 0.5))
```



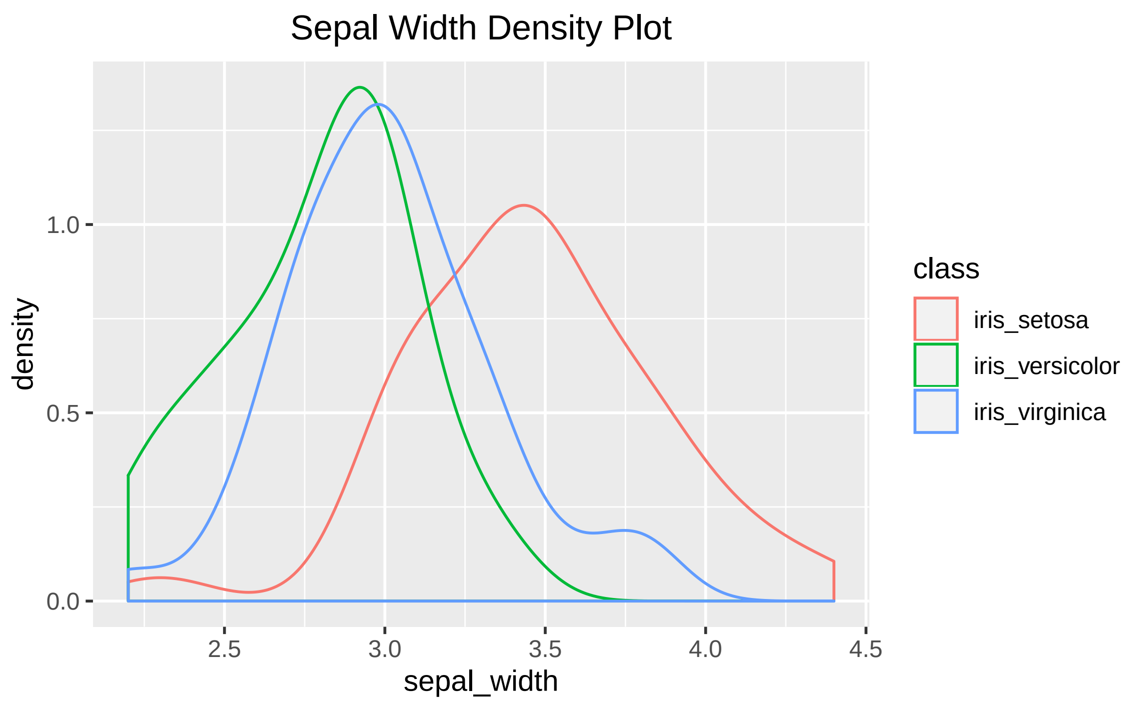
```r
ggplot(data = dataset, mapping = aes(x = class, y = petal_width)) +
  geom_boxplot() +
  ggtitle("Boxplots for Petal Width")+
  theme(plot.title = element_text(hjust = 0.5))
```



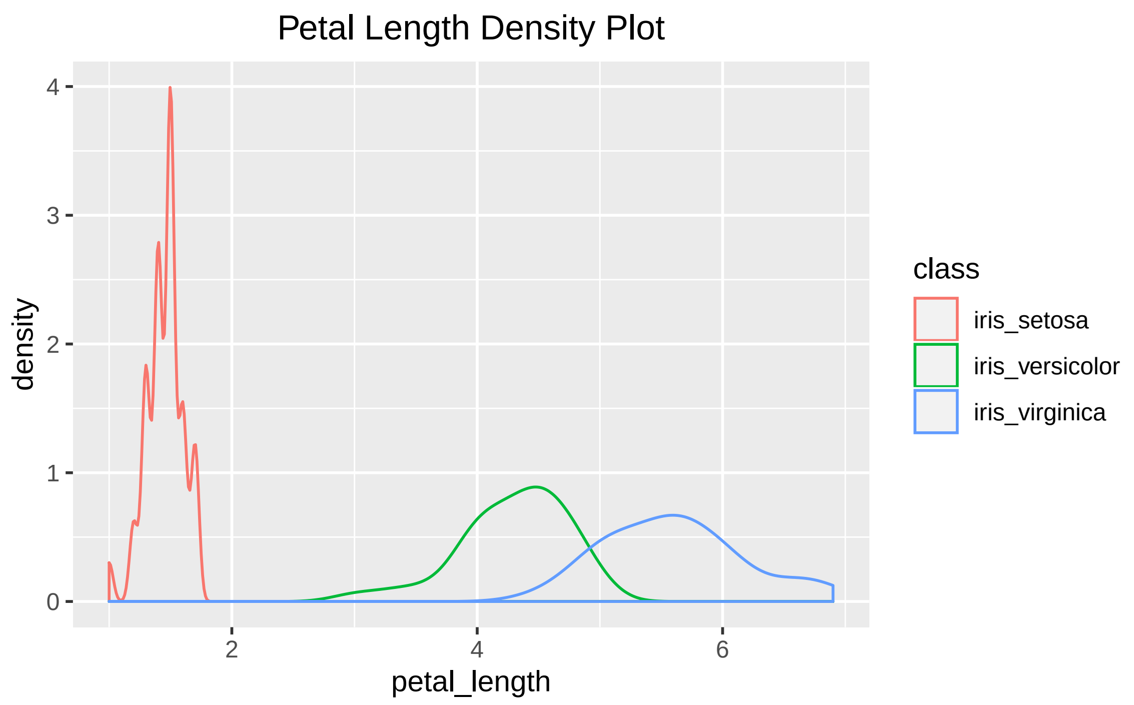
```
ggplot(data = dataset, mapping = aes(x = sepal_length, color = class)) +
  geom_density() +
  ggtitle("Sepal Lenth Density Plot")+
  theme(plot.title = element_text(hjust = 0.5))
```



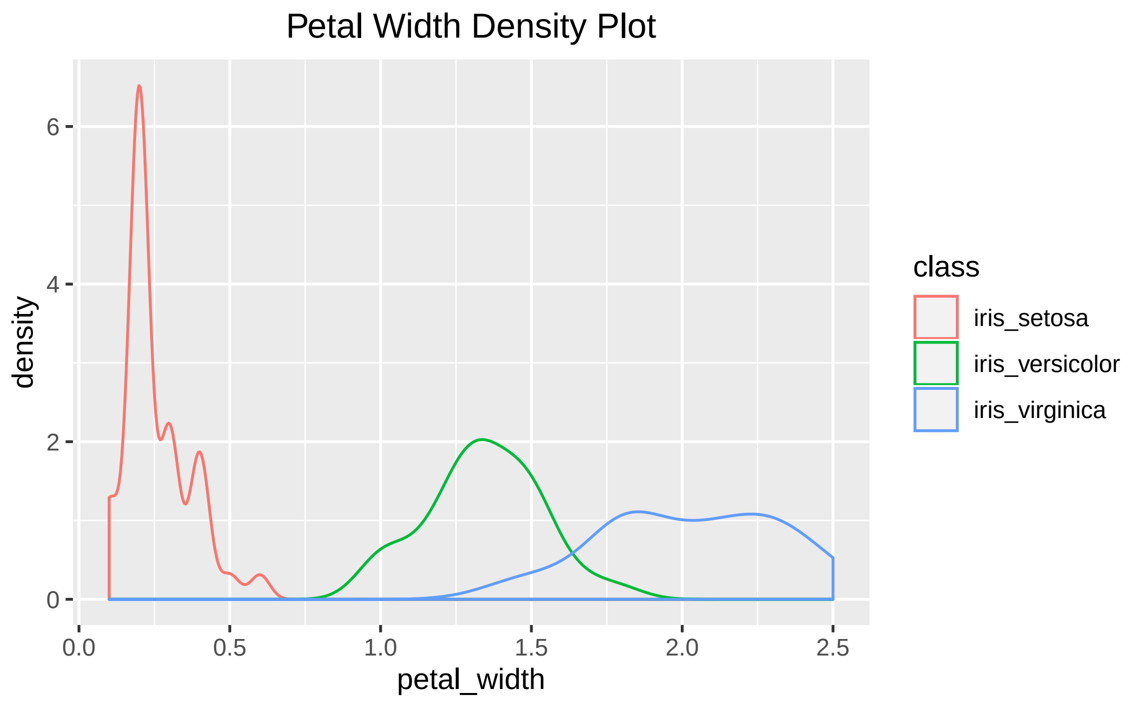
```
ggplot(data = dataset, mapping = aes(x = sepal_width, color = class)) +
  geom_density() +
  ggtitle("Sepal Width Density Plot")+
  theme(plot.title = element_text(hjust = 0.5))
```



```r
ggplot(data = dataset, mapping = aes(x = petal_length, color = class)) +
  geom_density() +
  ggtitle("Petal Length Density Plot")+
  theme(plot.title = element_text(hjust = 0.5))
```



```
ggplot(data = dataset, mapping = aes(x = petal_width, color = class)) +
  geom_density() +
  ggtitle("Petal Width Density Plot")+
  theme(plot.title = element_text(hjust = 0.5))
```



10-Fold Cross Validation and Model Building

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

To create a model that is actually a predictive model, use trainControl( ). In this case, the data will be split into 10 parts: 9 for training and 1 for validation.

```
# Run algorithms using 10-fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"
```

Now it is time to build models based on the data set. After visualizing the data, it appears a linear model would best represent the relationship between the variables. However, several models can be created and analyzed to see which model truly is the most accurate.

```
# a) linear algorithms
set.seed(8)
fit.lda <- train(class~., data=dataset, method="lda", metric=metric, trControl=control)
# b) nonlinear algorithms
```

```r
# CART
set.seed(8)
fit.cart <- train(class~., data=dataset, method="rpart", metric=metric, trControl=control)
# kNN
set.seed(8)
fit.knn <- train(class~., data=dataset, method="knn", metric=metric, trControl=control)
# c) advanced algorithms
# SVM
set.seed(8)
fit.svm <- train(class~., data=dataset, method="svmRadial", metric=metric, trControl=control)
# Random Forest
set.seed(8)
fit.rf <- train(class~., data=dataset, method="rf", metric=metric, trControl=control)
```

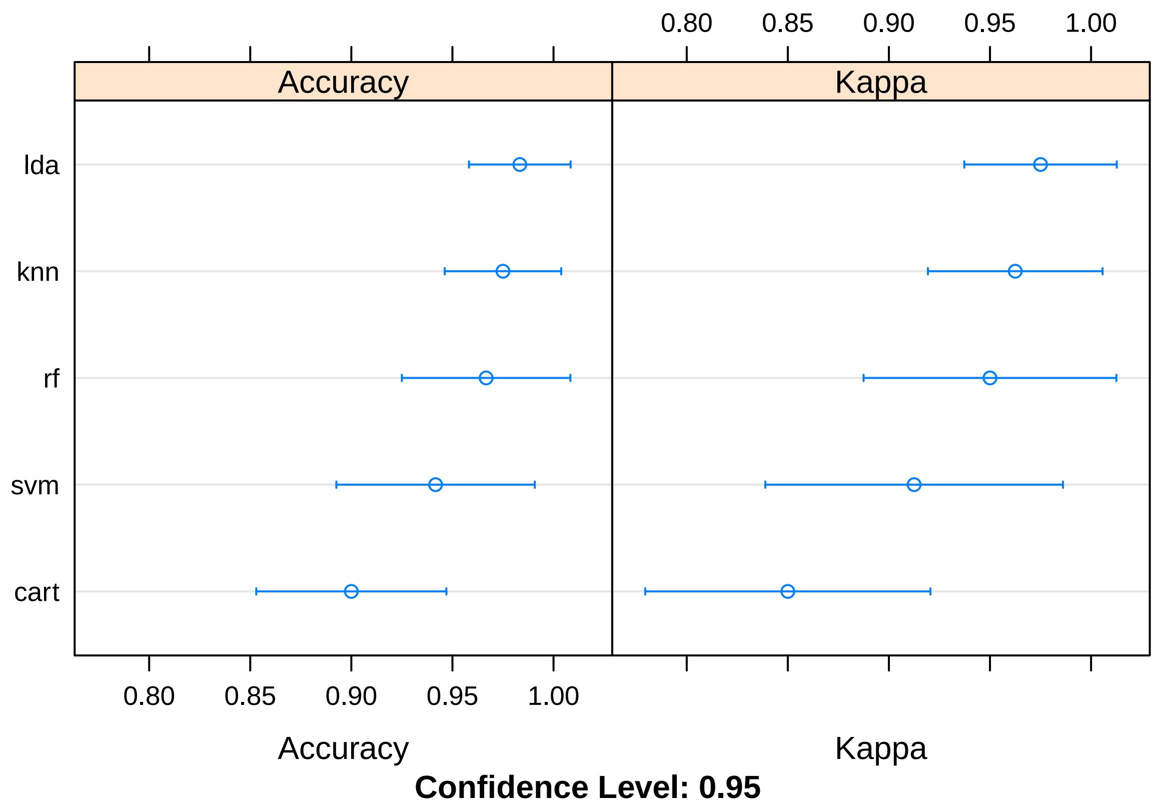
Run the code below to see the results of the models.

```r
# summarize accuracy of models
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## lda  0.9166667 1.0000000 1.0000000 0.9833333 1.0000000    1    0
## cart 0.8333333 0.8333333 0.9166667 0.9000000 0.9166667    1    0
## knn  0.9166667 0.9375000 1.0000000 0.9750000 1.0000000    1    0
## svm  0.8333333 0.9166667 0.9583333 0.9416667 1.0000000    1    0
## rf   0.8333333 0.9375000 1.0000000 0.9666667 1.0000000    1    0
##
## Kappa
##       Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
## lda  0.875 1.00000 1.0000 0.9750   1.000    1    0
## cart 0.750 0.75000 0.8750 0.8500   0.875    1    0
## knn  0.875 0.90625 1.0000 0.9625   1.000    1    0
## svm  0.750 0.87500 0.9375 0.9125   1.000    1    0
## rf   0.750 0.90625 1.0000 0.9500   1.000    1    0
```

Also, boxplots could be created to visualize the results.

```r
#box plot of the results
dotplot(results)
```



Based on the results from the models created, it appears the linear model (lda) is the most accurate predictive model.

```r
#Summarize the best model.
print(fit.lda)
```

```
## Linear Discriminant Analysis
##
## 120 samples
##   4 predictor
##   3 classes: 'iris_setosa', 'iris_versicolor', 'iris_virginica'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9833333  0.975
```

The linear model created has an accuracy of 98.3%.

Validation

Now it is time to see how the model holds up to data it has not seen before.

```r
# estimate skill of LDA on the validation dataset
predictions <- predict(fit.lda, validation)
confusionMatrix(predictions, validation$class)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        iris_setosa iris_versicolor iris_virginica
##   iris_setosa              10               0              0
##   iris_versicolor           0               9              0
##   iris_virginica            0               1             10
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95% CI : (0.8278, 0.9992)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 2.963e-13
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: iris_setosa Class: iris_versicolor
## Sensitivity                      1.0000                 0.9000
## Specificity                      1.0000                 1.0000
## Pos Pred Value                   1.0000                 1.0000
## Neg Pred Value                   1.0000                 0.9524
## Prevalence                       0.3333                 0.3333
## Detection Rate                   0.3333                 0.3000
## Detection Prevalence             0.3333                 0.3000
## Balanced Accuracy                1.0000                 0.9500
##                      Class: iris_virginica
## Sensitivity                         1.0000
## Specificity                         0.9500
## Pos Pred Value                      0.9091
## Neg Pred Value                      1.0000
## Prevalence                          0.3333
## Detection Rate                      0.3333
## Detection Prevalence                0.3667
## Balanced Accuracy                   0.9750
```

Conclusion

The results show the model missed identified one flower from the validation data set. The final test showed the final linear model predicted flower values at a 97% accuracy level. This performance is close to what the model showed during the training phase. While more time could be spent trying to improve the performance of the model, this report will conclude that the linear model created is sufficient.

Resources:

Three resources were used to complete the code used in this project.

1. “R for Data Science” by Garrett Grolemund and Hadley Wickham

- [link] (https://r4ds.had.co.nz/)

2. “Your First Machine Learning Project in R Step-By-Step” by Jason Brownlee

- [link] (https://machinelearningmastery.com/machine-learning-in-r-step-by-step/)

3. Iris Data Set

- [link] (https://archive.ics.uci.edu/ml/datasets/iris)