Application of Machine Learning Algorithms

Iris Yon
December 3, 2017

Table of Contents

Introduction

Machine learning is a popular field today that has been relevant for a long time. Given a particular data set, computational algorithms help train machines to autonomously predict models and results. From self driving cars to the movie recommendations on Netflix, machine learning is an emerging field becoming more relevant everyday.

Packages

Install Packages

intall.packages('caret', dependencies = TRUE) install.packages('ggplot2') install.packages('class')

Load all Libraries

```
# library with mathematical functions related to machine learning
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.2
```

```
# ggplot library
library(ggplot2)
# functions related to machine learning
library(class)
```

Data

The data used in this code is a built-in data set in R that can be accessed by typing "iris" in the console or code chunk. No additional packages are needed to be installed for the data source.

Iris Data Set

```
# data set
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
```

```
## 2           4.9         3.0          1.4         0.2     setosa
## 3           4.7         3.2          1.3         0.2     setosa
## 4           4.6         3.1          1.5         0.2     setosa
## 5           5.0         3.6          1.4         0.2     setosa
## 6           5.4         3.9          1.7         0.4     setosa
## 7           4.6         3.4          1.4         0.3     setosa
## 8           5.0         3.4          1.5         0.2     setosa
## 9           4.4         2.9          1.4         0.2     setosa
## 10          4.9         3.1          1.5         0.1     setosa
## 11          5.4         3.7          1.5         0.2     setosa
## 12          4.8         3.4          1.6         0.2     setosa
## 13          4.8         3.0          1.4         0.1     setosa
## 14          4.3         3.0          1.1         0.1     setosa
## 15          5.8         4.0          1.2         0.2     setosa
## 16          5.7         4.4          1.5         0.4     setosa
## 17          5.4         3.9          1.3         0.4     setosa
## 18          5.1         3.5          1.4         0.3     setosa
## 19          5.7         3.8          1.7         0.3     setosa
## 20          5.1         3.8          1.5         0.3     setosa
## 21          5.4         3.4          1.7         0.2     setosa
## 22          5.1         3.7          1.5         0.4     setosa
## 23          4.6         3.6          1.0         0.2     setosa
## 24          5.1         3.3          1.7         0.5     setosa
## 25          4.8         3.4          1.9         0.2     setosa
## 26          5.0         3.0          1.6         0.2     setosa
## 27          5.0         3.4          1.6         0.4     setosa
## 28          5.2         3.5          1.5         0.2     setosa
## 29          5.2         3.4          1.4         0.2     setosa
## 30          4.7         3.2          1.6         0.2     setosa
## 31          4.8         3.1          1.6         0.2     setosa
## 32          5.4         3.4          1.5         0.4     setosa
## 33          5.2         4.1          1.5         0.1     setosa
## 34          5.5         4.2          1.4         0.2     setosa
## 35          4.9         3.1          1.5         0.2     setosa
## 36          5.0         3.2          1.2         0.2     setosa
## 37          5.5         3.5          1.3         0.2     setosa
## 38          4.9         3.6          1.4         0.1     setosa
## 39          4.4         3.0          1.3         0.2     setosa
## 40          5.1         3.4          1.5         0.2     setosa
## 41          5.0         3.5          1.3         0.3     setosa
## 42          4.5         2.3          1.3         0.3     setosa
## 43          4.4         3.2          1.3         0.2     setosa
## 44          5.0         3.5          1.6         0.6     setosa
## 45          5.1         3.8          1.9         0.4     setosa
## 46          4.8         3.0          1.4         0.3     setosa
## 47          5.1         3.8          1.6         0.2     setosa
## 48          4.6         3.2          1.4         0.2     setosa
## 49          5.3         3.7          1.5         0.2     setosa
## 50          5.0         3.3          1.4         0.2     setosa
## 51          7.0         3.2          4.7         1.4 versicolor
## 52          6.4         3.2          4.5         1.5 versicolor
## 53          6.9         3.1          4.9         1.5 versicolor
## 54          5.5         2.3          4.0         1.3 versicolor
## 55          6.5         2.8          4.6         1.5 versicolor
## 56          5.7         2.8          4.5         1.3 versicolor
## 57          6.3         3.3          4.7         1.6 versicolor
## 58          4.9         2.4          3.3         1.0 versicolor
## 59          6.6         2.9          4.6         1.3 versicolor
## 60          5.2         2.7          3.9         1.4 versicolor
## 61          5.0         2.0          3.5         1.0 versicolor
## 62          5.9         3.0          4.2         1.5 versicolor
## 63          6.0         2.2          4.0         1.0 versicolor
## 64          6.1         2.9          4.7         1.4 versicolor
## 65          5.6         2.9          3.6         1.3 versicolor
## 66          6.7         3.1          4.4         1.4 versicolor
## 67          5.6         3.0          4.5         1.5 versicolor
## 68          5.8         2.7          4.1         1.0 versicolor
## 69          6.2         2.2          4.5         1.5 versicolor
## 70          5.6         2.5          3.9         1.1 versicolor
## 71          5.9         3.2          4.8         1.8 versicolor
## 72          6.1         2.8          4.0         1.3 versicolor
## 73          6.3         2.5          4.9         1.5 versicolor
## 74          6.1         2.8          4.7         1.2 versicolor
## 75          6.4         2.9          4.3         1.3 versicolor
## 76          6.6         3.0          4.4         1.4 versicolor
## 77          6.8         2.8          4.8         1.4 versicolor
## 78          6.7         3.0          5.0         1.7 versicolor
## 79          6.0         2.9          4.5         1.5 versicolor
## 80          5.7         2.6          3.5         1.0 versicolor
## 81          5.5         2.4          3.8         1.1 versicolor
```

```
## 82           5.5         2.4          3.7         1.0 versicolor
## 83           5.8         2.7          3.9         1.2 versicolor
## 84           6.0         2.7          5.1         1.6 versicolor
## 85           5.4         3.0          4.5         1.5 versicolor
## 86           6.0         3.4          4.5         1.6 versicolor
## 87           6.7         3.1          4.7         1.5 versicolor
## 88           6.3         2.3          4.4         1.3 versicolor
## 89           5.6         3.0          4.1         1.3 versicolor
## 90           5.5         2.5          4.0         1.3 versicolor
## 91           5.5         2.6          4.4         1.2 versicolor
## 92           6.1         3.0          4.6         1.4 versicolor
## 93           5.8         2.6          4.0         1.2 versicolor
## 94           5.0         2.3          3.3         1.0 versicolor
## 95           5.6         2.7          4.2         1.3 versicolor
## 96           5.7         3.0          4.2         1.2 versicolor
## 97           5.7         2.9          4.2         1.3 versicolor
## 98           6.2         2.9          4.3         1.3 versicolor
## 99           5.1         2.5          3.0         1.1 versicolor
## 100          5.7         2.8          4.1         1.3 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 102          5.8         2.7          5.1         1.9  virginica
## 103          7.1         3.0          5.9         2.1  virginica
## 104          6.3         2.9          5.6         1.8  virginica
## 105          6.5         3.0          5.8         2.2  virginica
## 106          7.6         3.0          6.6         2.1  virginica
## 107          4.9         2.5          4.5         1.7  virginica
## 108          7.3         2.9          6.3         1.8  virginica
## 109          6.7         2.5          5.8         1.8  virginica
## 110          7.2         3.6          6.1         2.5  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 112          6.4         2.7          5.3         1.9  virginica
## 113          6.8         3.0          5.5         2.1  virginica
## 114          5.7         2.5          5.0         2.0  virginica
## 115          5.8         2.8          5.1         2.4  virginica
## 116          6.4         3.2          5.3         2.3  virginica
## 117          6.5         3.0          5.5         1.8  virginica
## 118          7.7         3.8          6.7         2.2  virginica
## 119          7.7         2.6          6.9         2.3  virginica
## 120          6.0         2.2          5.0         1.5  virginica
## 121          6.9         3.2          5.7         2.3  virginica
## 122          5.6         2.8          4.9         2.0  virginica
## 123          7.7         2.8          6.7         2.0  virginica
## 124          6.3         2.7          4.9         1.8  virginica
## 125          6.7         3.3          5.7         2.1  virginica
## 126          7.2         3.2          6.0         1.8  virginica
## 127          6.2         2.8          4.8         1.8  virginica
## 128          6.1         3.0          4.9         1.8  virginica
## 129          6.4         2.8          5.6         2.1  virginica
## 130          7.2         3.0          5.8         1.6  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 133          6.4         2.8          5.6         2.2  virginica
## 134          6.3         2.8          5.1         1.5  virginica
## 135          6.1         2.6          5.6         1.4  virginica
## 136          7.7         3.0          6.1         2.3  virginica
## 137          6.3         3.4          5.6         2.4  virginica
## 138          6.4         3.1          5.5         1.8  virginica
## 139          6.0         3.0          4.8         1.8  virginica
## 140          6.9         3.1          5.4         2.1  virginica
## 141          6.7         3.1          5.6         2.4  virginica
## 142          6.9         3.1          5.1         2.3  virginica
## 143          5.8         2.7          5.1         1.9  virginica
## 144          6.8         3.2          5.9         2.3  virginica
## 145          6.7         3.3          5.7         2.5  virginica
## 146          6.7         3.0          5.2         2.3  virginica
## 147          6.3         2.5          5.0         1.9  virginica
## 148          6.5         3.0          5.2         2.0  virginica
## 149          6.2         3.4          5.4         2.3  virginica
## 150          5.9         3.0          5.1         1.8  virginica
```

```
# column names
colnames(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```
# number of columns = 5
ncol(iris)
```

```
## [1] 5
```

```
# number of rows = 150
nrow(iris)
```

```
## [1] 150
```

```
# summary statistics for data
summary(iris)
```

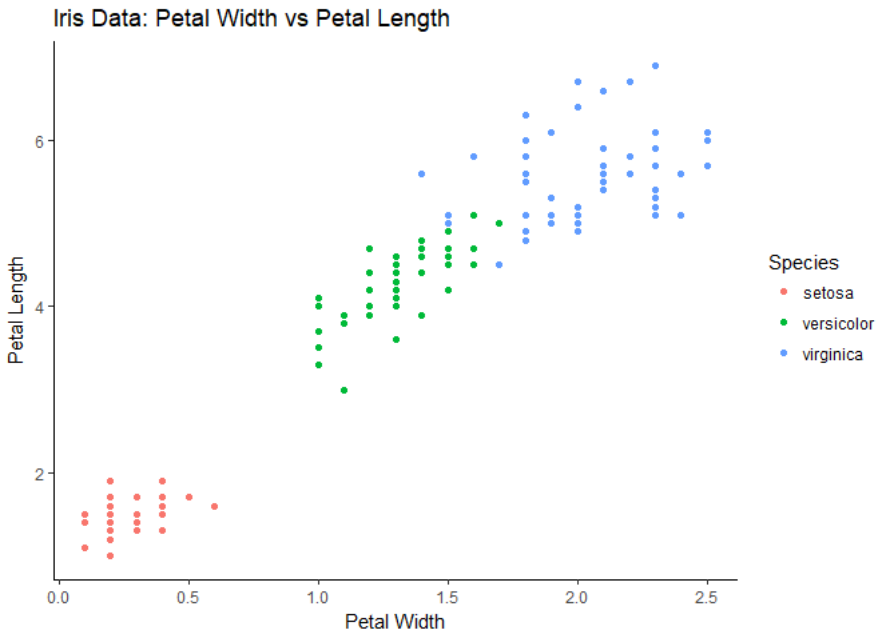
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
# structure of data
str(data)
```

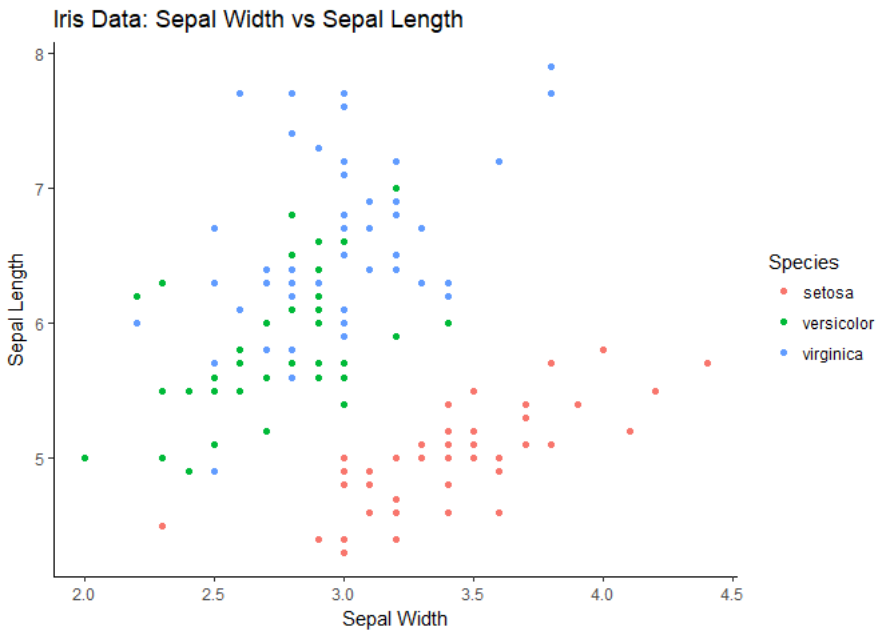
```
## function (..., list = character(), package = NULL, lib.loc = NULL,
##     verbose = getOption("verbose"), envir = .GlobalEnv)
```

Visualizing the Data

```
ggplot(iris, aes(Petal.Width, Petal.Length, color = Species)) +
    geom_point() + ggtitle("Iris Data: Petal Width vs Petal Length") + theme_classic() + labs(x = "Petal Wi
```



```
ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +
    geom_point() + ggtitle("Iris Data: Sepal Width vs Sepal Length") + theme_classic() + labs(x = "Sepal Wi
```



Algorithms

Random Number Generator to make Training Sets & Test Sets

```
set.seed(111)
```

The training set refers to the data set the machine is trained on to predict the most accurate results tested against the test sets that have the desired results. In this case, we take a portion of the training and test set from the iris data set in order to predict if the machine can accurately output the iris flower species based on given inputs for the sepal width, sepal length, petal width, adn petal length.

Creating the Training and Test Sets

```
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.75, .25))

# Training Set
iris.training <- iris[ind==1, 1:4]

# Test Set
iris.testing <- iris[ind==2, 1:4]

iris.trainingPrediction <- iris[ind==1, 5]

iris.testingPrediction <- iris[ind==2, 5]
```

Testing the different probability values shows that increasing the value for the training set (0.75) outputs more accurate results from the machine.

K Nearest Neighbors Algorithm

```
prediction <- knn(train = iris.training, test = iris.testing, cl = iris.trainingPrediction, k=3)
```

This algorithm uses the Euclidian distance measure to identify inputs and categorize them into the same group based on similar characteristics. We use this algorithm to predict the flower species based on the inputs we get from the training set.

Compare Results

```
compare_results <- function(train, test) {
  count <- 0
  for (i in 1:nrow(test)) {
    training <- as.character(train[i, ])
    testing <- as.character(test[i, ])
    if (training != testing) {
      cat(sprintf("Prediction = \"%s\" | Actual =  \"%s\"\n", training, testing))
      count <- count + 1
    }
  }
  cat(sprintf("\n%i Occurences of Mismatch", count))
}

test <- data.frame(iris.testingPrediction)
train <- data.frame(prediction)
compare_results(train, test)
```

```
## Prediction = "versicolor" | Actual =  "virginica"
##
## 1 Occurences of Mismatch
```

The K Nearest Neighbors algorithm gives a relatively accurate predictive model for guessing the flower species given a number of training samples.

Make Data for Caret Package Functions

```
set.seed(111)
# get index of species to predict and probability of training set inputs to testing set
index <- createDataPartition(iris$Species, p=0.75, list=FALSE)

# training set
iris.trainingCaret <- iris[index,]

# testing set
iris.testingCaret <- iris[-index,]
```

Using K Nearest Neighbors Algorithm from Caret Package

```
model_knn <- train(iris.trainingCaret[, 1:4], iris.trainingCaret[, 5], method='knn', preProcess=c("center",

# Predict values
predictions<-predict.train(object=model_knn,iris.testingCaret[,1:4], type="raw")

# Confusion matrix showing predictions compared to test set and overall statistics
confusionMatrix(predictions,iris.testingCaret[,5])
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         12         2
##   virginica       0          0        10
##
## Overall Statistics
##
##                Accuracy : 0.9444
##                  95% CI : (0.8134, 0.9932)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 1.728e-14
##
##                   Kappa : 0.9167
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.8333
## Specificity                 1.0000            0.9167           1.0000
## Pos Pred Value              1.0000            0.8571           1.0000
## Neg Pred Value              1.0000            1.0000           0.9231
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.2778
## Detection Prevalence        0.3333            0.3889           0.2778
## Balanced Accuracy           1.0000            0.9583           0.9167
```

Using Random Forest Algorithm from Caret Package

```
model_rf <- train(iris.trainingCaret[, 1:4], iris.trainingCaret[, 5], method='rf', preProcess=c("center", '
```

```
## Warning: package 'randomForest' was built under R version 3.4.3

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

```
# Predict values
predictions_rf <-predict.train(object=model_rf,iris.testingCaret[,1:4], type="raw")

# Confusion matrix showing predictions compared to test set and overall statistics
confusionMatrix(predictions_rf, iris.testingCaret[,5])
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         11         2
##   virginica       0          1        10
##
## Overall Statistics
##
##                Accuracy : 0.9167
##                  95% CI : (0.7753, 0.9825)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 3.978e-13
##
##                   Kappa : 0.875
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9167           0.8333
## Specificity                 1.0000            0.9167           0.9583
## Pos Pred Value              1.0000            0.8462           0.9091
## Neg Pred Value              1.0000            0.9565           0.9200
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3056           0.2778
## Detection Prevalence        0.3333            0.3611           0.3056
## Balanced Accuracy           1.0000            0.9167           0.8958
```

The Random Forest Algorithm uses the idea of tree recursion found in computer science. Given a list of classifiers and numerical ranges, the random forest algorithm takes the features of each input and calculates its outcome, then it compares it with the best fit test result to predict its final output.

Using Support Vector Machine Algorithm from Caret Package

```
model_svm <- train(iris.trainingCaret[, 1:4], iris.trainingCaret[, 5], method='svmRadial')
```

```
##
## Attaching package: 'kernlab'

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

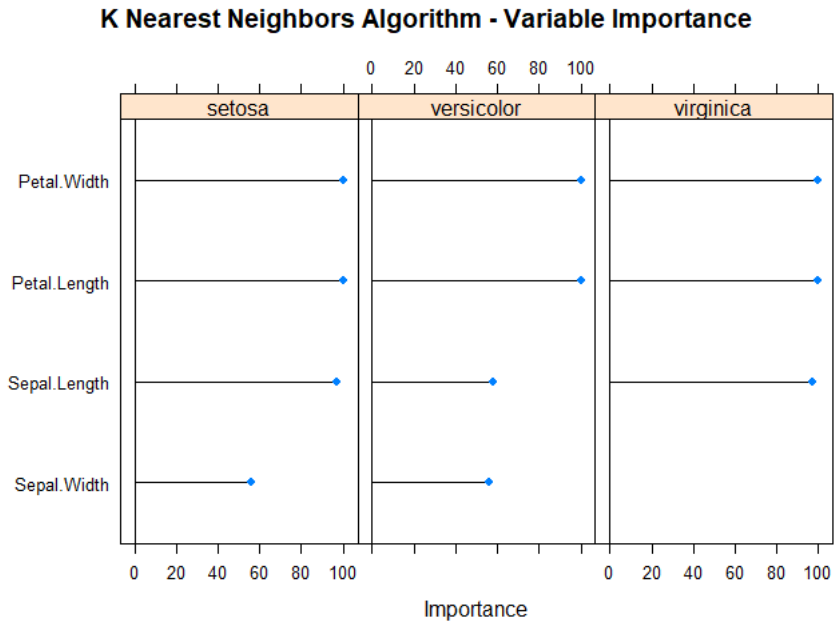
```
# Predict values
predictions_svm <-predict.train(object=model_svm, iris.testingCaret[,1:4], type="raw")

# Confusion matrix showing predictions compared to test set and overall statistics
confusionMatrix(predictions_svm, iris.testingCaret[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0         11         1
##   virginica       1          1        11
##
## Overall Statistics
##
##                Accuracy : 0.9167
##                  95% CI : (0.7753, 0.9825)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 3.978e-13
##
##                   Kappa : 0.875
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 0.9167            0.9167           0.9167
## Specificity                 1.0000            0.9583           0.9167
## Pos Pred Value              1.0000            0.9167           0.8462
## Neg Pred Value              0.9600            0.9583           0.9565
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3056            0.3056           0.3056
## Detection Prevalence        0.3056            0.3333           0.3611
## Balanced Accuracy           0.9583            0.9375           0.9167
```

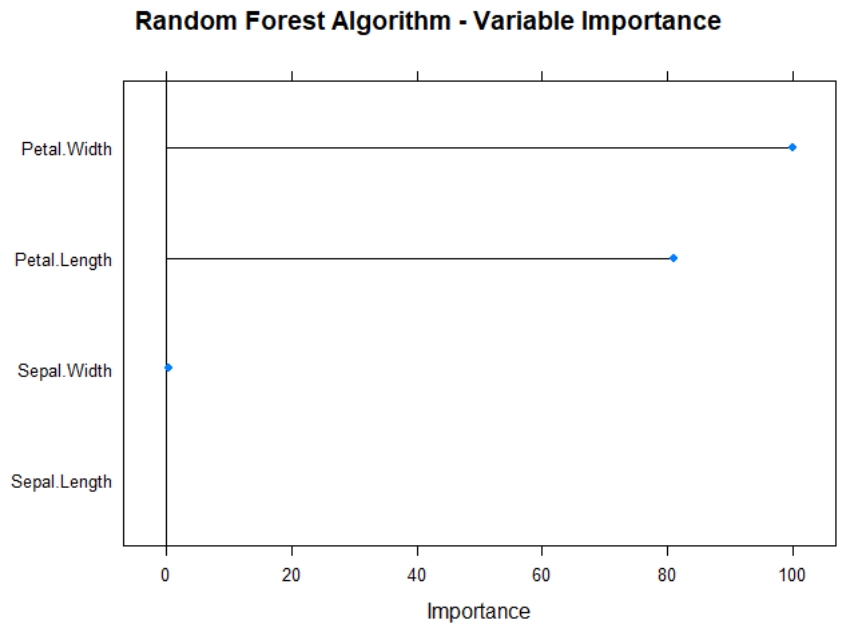
The Support Vector Machine Algorithm plots the given inputs based on their characteristics and calculates the hyper plane that separates two groups. Based on the location of the plotted input, the inputs are categorized into a predicted group and returned as the output.

Overview of Results and Algorithms

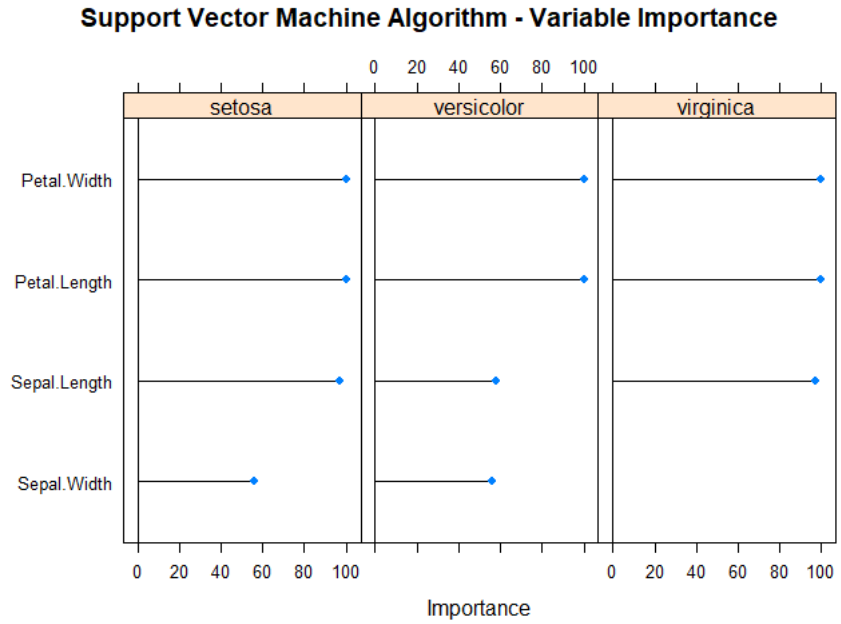
```
plot(varImp(object=model_knn),main="K Nearest Neighbors Algorithm - Variable Importance")
```




```
plot(varImp(object=model_rf),main="Random Forest Algorithm - Variable Importance")
```




```
plot(varImp(object=model_svm),main="Support Vector Machine Algorithm – Variable Importance")
```



It is interesting to note that the variable dependencies for the K Nearest Neighbors and Support Vector Machine Algorithm are most similar, although the Support Vector Machine Algorithm matches the accuracy of the Random Forest Algorithm. The K Nearest Neighbors Algorithm has the highest accuracy, with the other two algorithms coming second.

Discussion & Message

Using data packages such as class and caret, we analyze the built-in data set under "iris" to evaluate whether the machine can predict the species of the iris flower when given inputs for sepal length, sepal width, petal width, and petal length.

Based on the outcomes, the machine sufficiently predicts the flower species based on the training set and compared with the testing test created with the original iris data set. We closely studied the K Nearest Neighbor Algorithm, and tested the accuracy of the results compared to the Random Forest Algorithm and the Support Vector Machine Algorithm.

Among the 3 different types of machine learning algorithms, the K Nearest Neighbor Algorithm proved to be the most accurate. The Random Forest Algorithm and Vector Machine Algorithm both had the same accuracy, although they predicted different species when compared against the actual testing set.

Conclusion

It was interesting to explore the topic of machine learning in the R environment given its many tools to use plotting and other data visualization tools to model data sets. Looking into different machine learning algorithms, the importance of data and its applications became increasingly evident. It was fun to see examples where advanced technology was founded on applied mathematics in its application of approximations and conceptual models, and gives more reason to value the material learned in this class.

References

1. Machine Learning Background
   - background, definition, and examples of how machine learning
1. Caret Library
   - library with functions that allow for creation of predictive models
1. Data Visualization Packages
   - article about best libraries and packages for data visualizations in R
   - includes uses, examples, short description of each R package
1. Machine Learning Project Ideas
   - show how machine learning techniques can be applied to various types of data
1. K Nearest Neighbors
   - explanation of K Nearest Neighbors algorithm
1. Random Forest Algorithm
   - explanation of the Random Forest Algorithm and its applications
1. Support Vector Machine Algorithm
   - explanation of the Support Vector Machine Algorithm in machine learning
1. Different Machine Learning Functions in Caret Package
   - outlines the many functions used for classification or regression of data

---