Machine Learning in R

Rose Condon

6/21/2020

Introduction

This excerise is based on the blog written by Jason Brownlee "Spot Check Machine Learning Algorithms in R". The blog has great insights for planning Machine Learning project, regardless the project written in any programming language.

This excerise contain steps details of data anlysis with machine learning from R package.

Implementation

Clean environment

```
rm(list = ls())
```

Load required

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

Load data, we are using the iris datasets from caret package

```
data(iris)
dataset <- iris
# view datasets, you can use either dataset or iris, they are identical
head(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
```

Data preparation

```
# Create a sample from original dataset, split sample data 80% for training model and validation
sample_index <- createDataPartition(dataset$Species, p=0.80, list=FALSE)
validation <- dataset[-sample_index,] # 20% of oroginal
# view validation datasets
tail(validation)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 119          7.7         2.6          6.9         2.3 virginica
## 123          7.7         2.8          6.7         2.0 virginica
## 131          7.4         2.8          6.1         1.9 virginica
## 132          7.9         3.8          6.4         2.0 virginica
## 142          6.9         3.1          5.1         2.3 virginica
## 143          5.8         2.7          5.1         1.9 virginica
```

```
# 80% for training data
t_dataset <- dataset[sample_index,]
# view training datasets
head(t_dataset)
```

```
##   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
## 6          5.4         3.9          1.7         0.4  setosa
## 7          4.6         3.4          1.4         0.3  setosa
```

Here, we do some EDA

```
# Dimension
dim(t_dataset)
```

```
## [1] 120   5
```

```
# check type of attributes (also called predictors)
sapply(t_dataset, class)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

```
# A class variable type is a factor which has multiple class labels or leverls. Here we have Species va
levels(t_dataset$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

Now we know in this dataset, we can refer to an attribute by name / labels as a property of the dataset. [1] “setosa” “versicolor” “virginica”

A typical multi-class or a multinomial classification problem.

Before we work at this classification problem, let's have a look at the class distribution

```
percentage <- prop.table(table(t_dataset$Species)) * 100
# Create a dataset to have only our factor and distribution
cbind(freq=table(t_dataset$Species), percentage=percentage)
```

```
##            freq percentage
## setosa       40   33.33333
## versicolor   40   33.33333
## virginica    40   33.33333
```

```
# View training dataset statistical summary
summary(t_dataset)
```

```
##   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 :5.750   Median :3.000   Median :4.300   Median :1.300
##  Mean   :5.797   Mean   :3.059   Mean   :3.723   Mean   :1.193
##  3rd Qu.:6.325   3rd Qu.:3.325   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.700   Max.   :4.400   Max.   :6.700   Max.   :2.500
##        Species
##  setosa    :40
##  versicolor:40
##  virginica :40
##
##
##
```

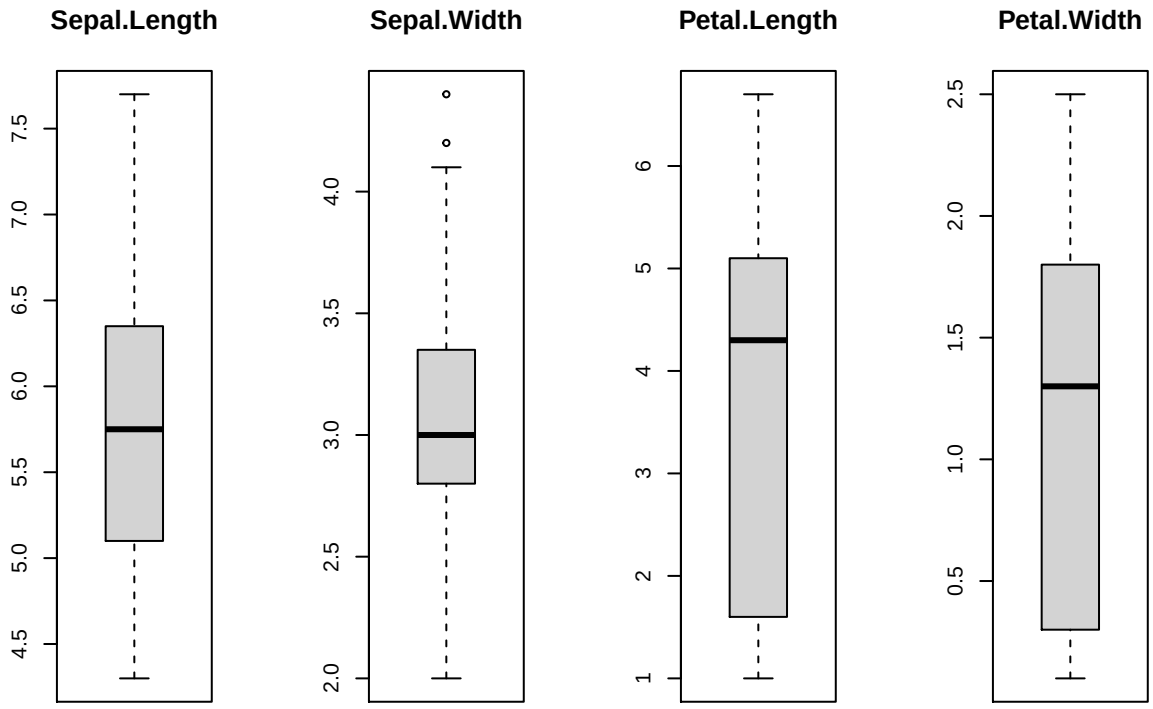
Plots of individual variable

set up x and y for input and output

```
x <- t_dataset[,1:4] # col 1-4
y <- t_dataset[,5]  # col 5 => Species
```

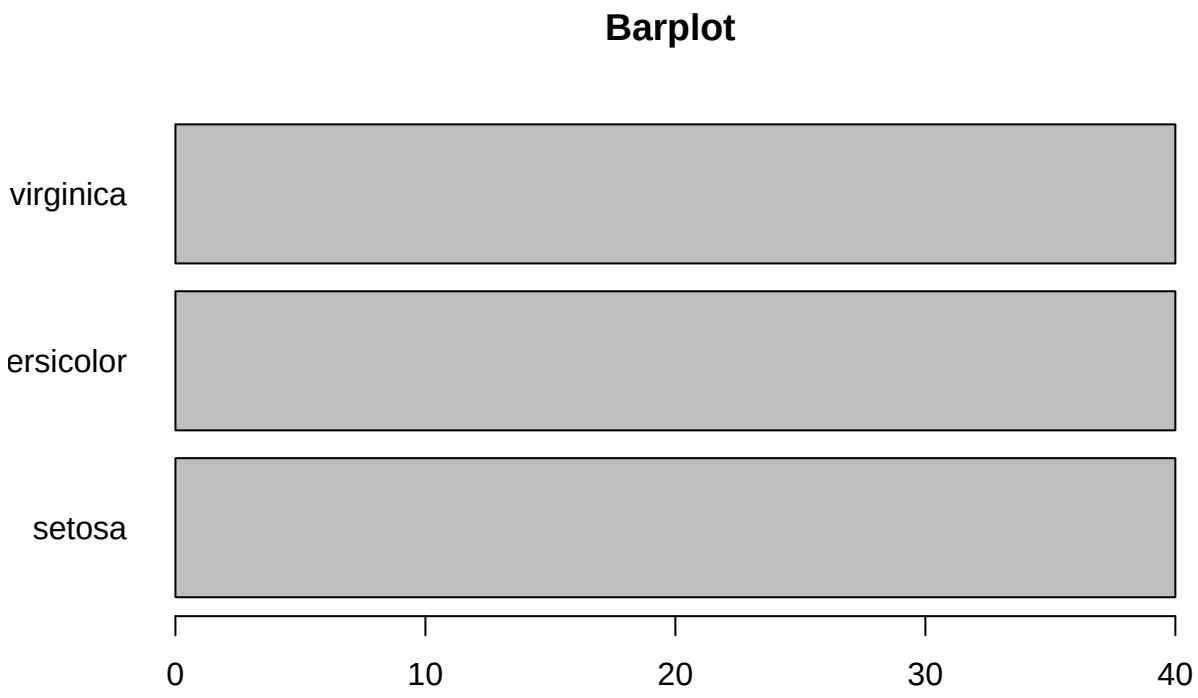
Boxplot for each attri (col 1-4)

```
par(mfrow=c(1,4))
for(i in 1:4) {
  boxplot(x[,i], main=names(iris)[i])
}
```



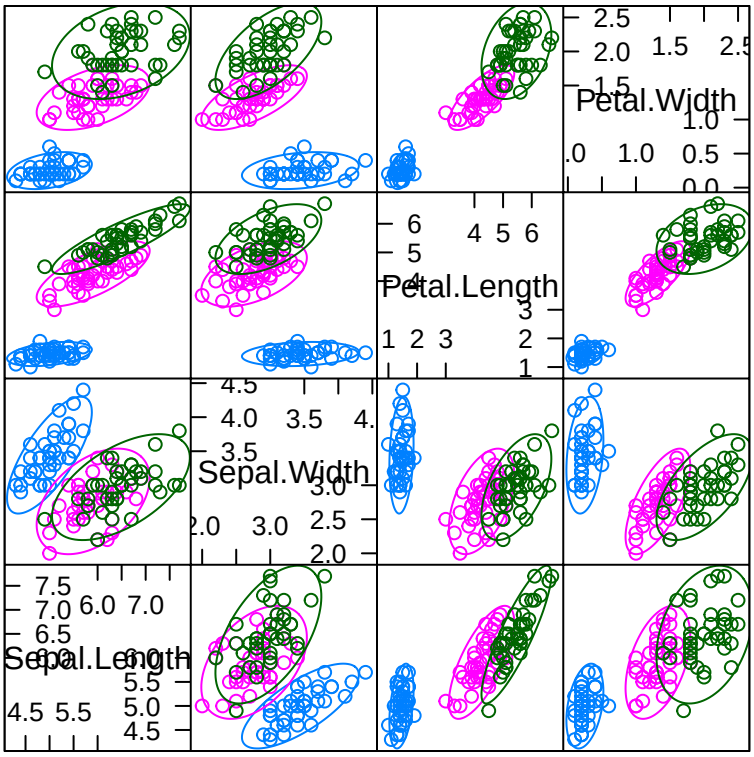
Barplot for class (Species)

```
plot(y, main="Barplot", las=1, horiz=TRUE)
```



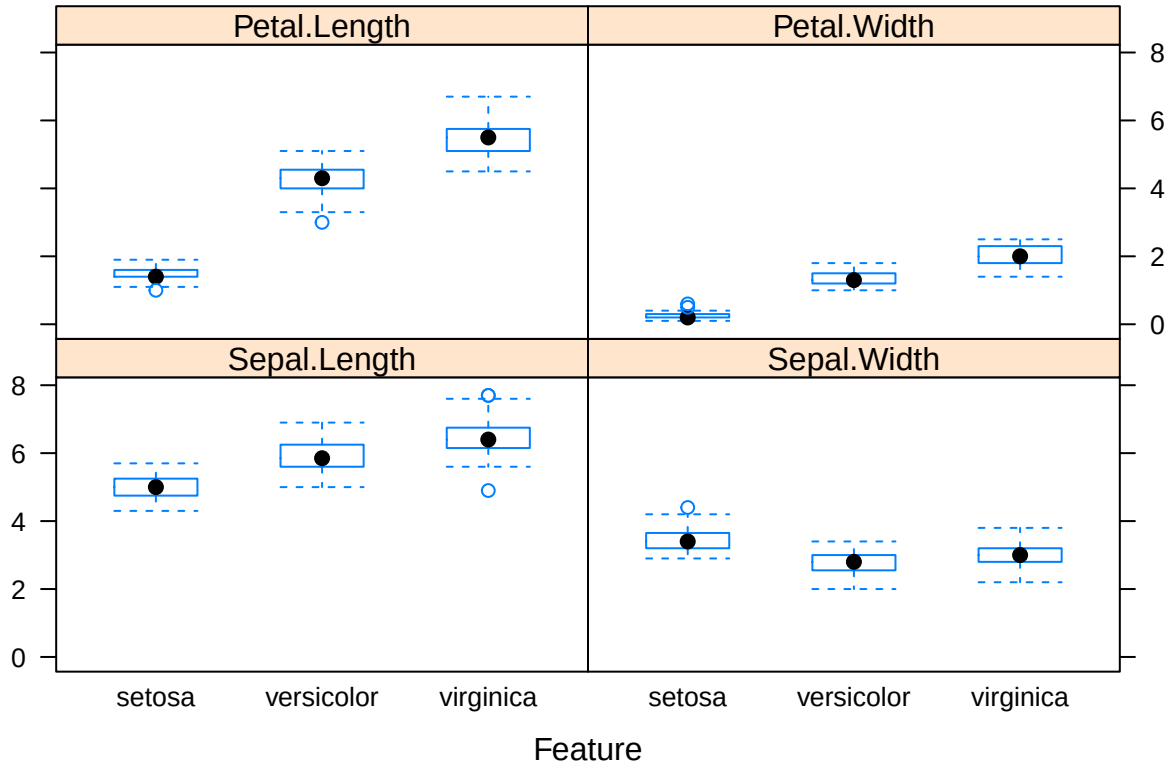
Scatter Plots to look at the interaction between the variables

```
# Note: You may need to run install.packages("ellipse")
featurePlot(x=x, y=y, plot="ellipse")
```



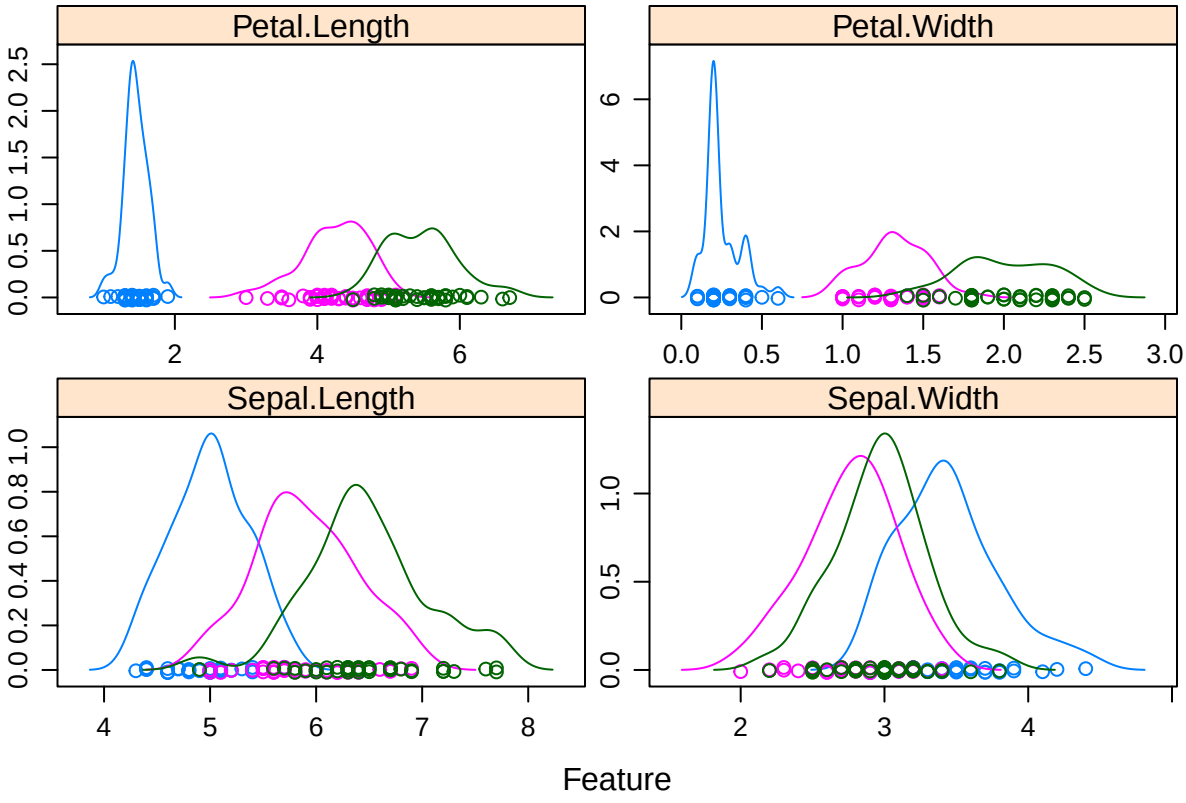
Scatter Plot Matrix

```
# and Box plot
featurePlot(x=x, y=y, plot="box")
```



Density plots for each attribute by class value

```
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```



Build our ML models

We will build model with k-folder cross-validation, using trainControl function to generate params to control how model will be built.

```
k<-10
control <- trainControl(method="cv", number=10)
# will use the metric of "Accuracy" to evaluate models
metric <- "Accuracy"
```

These are popular ML algorithms we will use to build ML models

```
# a) Linear Discriminant Analysis (LDA)
set.seed(1)
m_lda <- train(Species~., data=t_dataset, method="lda", metric=metric, trControl=control)
# b) Classification and Regression Trees (CART)
set.seed(1)
m_cart <- train(Species~., data=t_dataset, method="rpart", metric=metric, trControl=control)
# c) KNN - nonlinear
set.seed(1)
m_knn <- train(Species~., data=t_dataset, method="knn", metric=metric, trControl=control)
# d) SVM with a linear kernel
set.seed(1)
```

```
m_svm <- train(Species~., data=t_dataset, method="svmRadial", metric=metric, trControl=control)
# e) Random Forest
set.seed(1)
m_rf <- train(Species~., data=t_dataset, method="rf", metric=metric, trControl=control)
```

Estimate all 5 models performance by checking accuracy

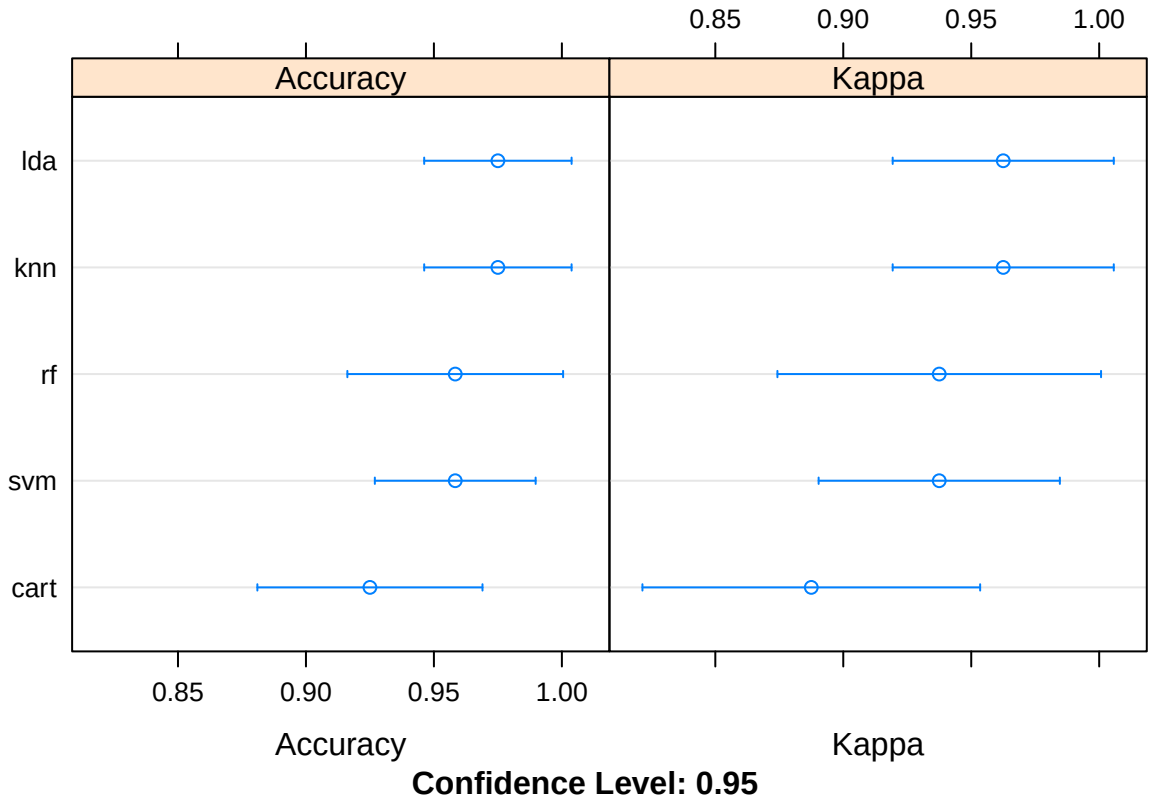
```
accuracies <- resamples(list(lda=m_lda, cart=m_cart, knn=m_knn, svm=m_svm, rf=m_rf))
summary(accuracies)
```

```
##
## Call:
## summary.resamples(object = accuracies)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## lda  0.9166667 0.9375000 1.0000000 0.9750000 1.0000000    1    0
## cart 0.8333333 0.9166667 0.9166667 0.9250000 0.9791667    1    0
## knn  0.9166667 0.9375000 1.0000000 0.9750000 1.0000000    1    0
## svm  0.9166667 0.9166667 0.9583333 0.9583333 1.0000000    1    0
## rf   0.8333333 0.9166667 1.0000000 0.9583333 1.0000000    1    0
##
## Kappa
##       Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
## lda  0.875 0.90625 1.0000 0.9625 1.00000    1    0
## cart 0.750 0.87500 0.8750 0.8875 0.96875    1    0
## knn  0.875 0.90625 1.0000 0.9625 1.00000    1    0
## svm  0.875 0.87500 0.9375 0.9375 1.00000    1    0
## rf   0.750 0.87500 1.0000 0.9375 1.00000    1    0
```

We are looking for accuracy more close to 1.0

Plot them

```
dotplot(accuracies)
```



So, from accuracy and plot, we know LDA model is more accurate.

Double check LDA model details

```
print(m_lda)
```

```
## Linear Discriminant Analysis
##
## 120 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.975     0.9625
```

You may also run str(m_lda) to find accuracy SD

LDA model has 98% accuracy and accuracy +/- 3%.

Make prediction

We built LDA model, now we should find out the accuracy of the model on our validation dataset.

```
predictions <- predict(m_lda, validation)
confusionMatrix(predictions, validation$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          0        10
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.8843, 1)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 4.857e-15
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           1.0000
## Specificity                 1.0000            1.0000           1.0000
## Pos Pred Value              1.0000            1.0000           1.0000
## Neg Pred Value              1.0000            1.0000           1.0000
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.3333
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            1.0000           1.0000
```

The accuracy for validation dataset is 96.7%, close what it predicts on traing dataset which is 98%

So, we can say our **LDA** ML model is accurate and a reliable for this classification problem.