Machine Learning & Iris Flower Data

LvdT

Introduction

This project will investigate *Fisher's Iris flower data set* through various machine learning algorithms, such as linear discriminant analysis, classification and regression trees, k-nearest neighbours, support vector machine and random forest. We will attempt to train a model to classify the flower's species correctly.

Packages and data import

The required packages and imports are as follows:

```
library(caret)
library(lattice)
library(ggplot2)
source("QuickCairoExport.R")
```

The data is imported as follows:

```
hardsource = FALSE

if (hardsource == TRUE) {
  source <- "iris.csv"
  dataset <- read.csv(source, header = FALSE)
  colnames(dataset) <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species")
}

dataset <- iris
```

Partitioning

In preparation for the machine learning algorithms later on, the data is split into test and validation partitions here.

```
# Create validation set
validation_index <- createDataPartition(dataset$Species, p = 0.8, list = FALSE)
validation <- dataset[-validation_index, ]

# Create testing set
dataset <- dataset[validation_index, ]
```

Basic data exploration

This section will briefly explore the data set.

```
dim(dataset)
```

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

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

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

```
head(dataset, n = 8)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 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
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 11          5.4         3.7          1.5         0.2  setosa
```

The species variable

As can be seen from above, the training data set contains 120 observations of 5 variables. The first four of these are numeric. The last one, flower species, is categorical. The different categories for this factorised variable can be seen below, along with the distribution among different types of species within the variable.

```
levels(dataset$Species)
```

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

```
# Class distribution
percentage <- prop.table(table(dataset$Species)) * 100
frequency <- table(dataset$Species)
cbind(frequency, percentage)
```

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

General summary statistics for the training data set are as follows:

```
summary(dataset)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.400   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 :4.200   Median :1.300
##  Mean   :5.847   Mean   :3.042   Mean   :3.739   Mean   :1.193
##  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    :40
##  versicolor:40
##  virginica :40
##
##
##
```

Visualisation

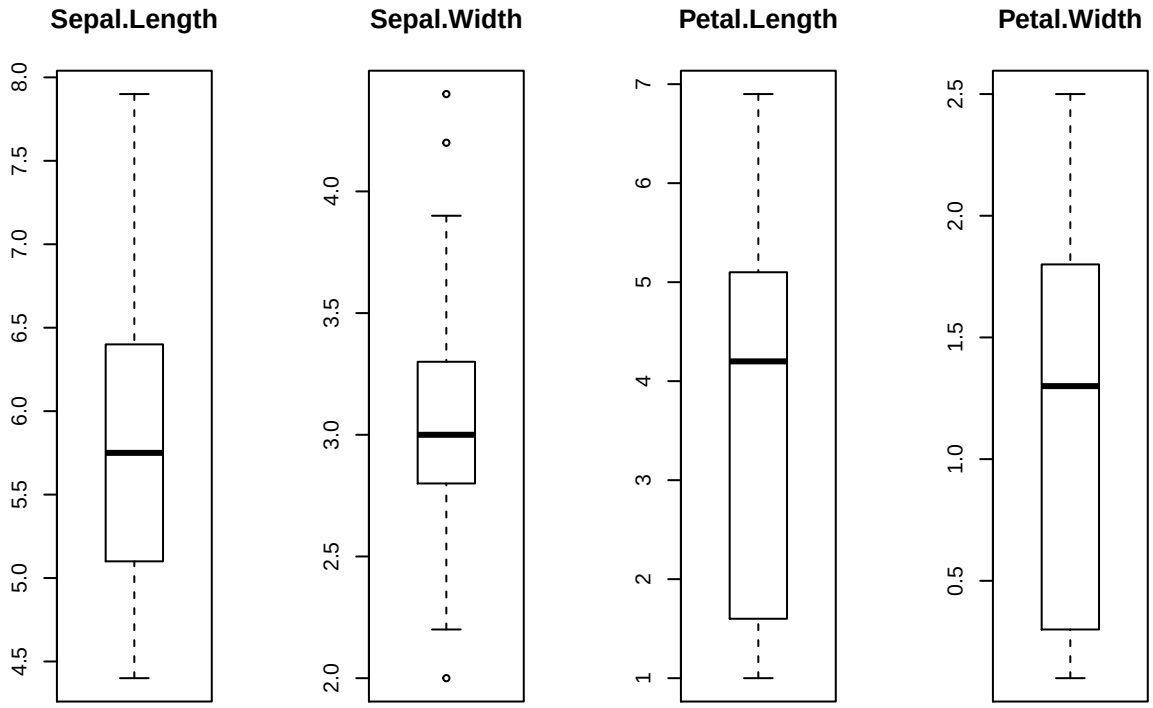
Graphics often tell the story a lot more smoothly than numbers alone. This section will present some visualisations. However, first we need to separate the factorised variable from the numeric ones.

```
# Split by class; numeric and factors
x <- dataset[, 1:4]
y <- dataset[, 5]
```

Boxplots

Univariate boxplots for the numerical variables:

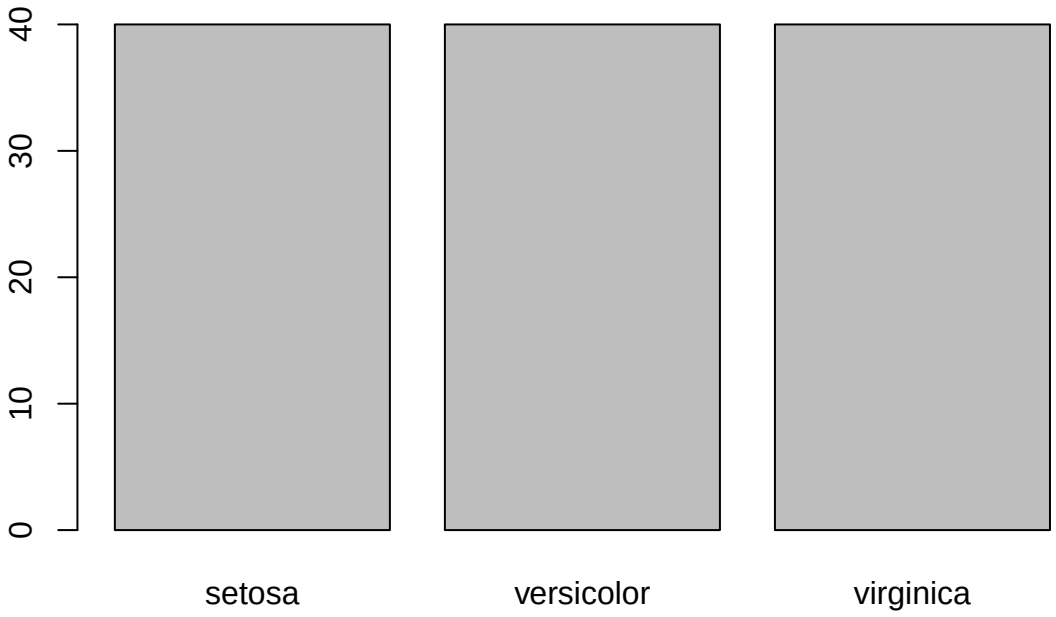
```
par(mfrow = c(1, 4))
  for(i in 1:4) {
    boxplot(x[, i], main = names(dataset)[i])
}
```



Barplot

Univariate barplot for the factorised variable:

```
par(mfrow = c(1, 1))
plot(y)
```

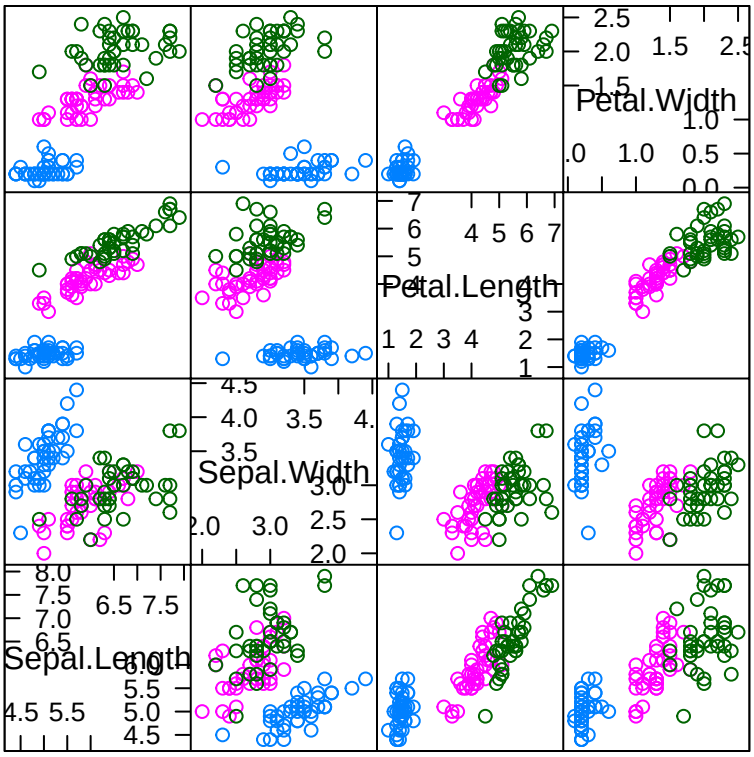


Multivariate plots

Multivariate scatterplot

The following scatterplot matrix clearly shows some relationships between predictor variables (along axes) and classes (the species by colours).

```
# pairs(dataset[ ,-5], col = dataset$Species)
featurePlot(x = x, y = y, plot = "pairs")
```

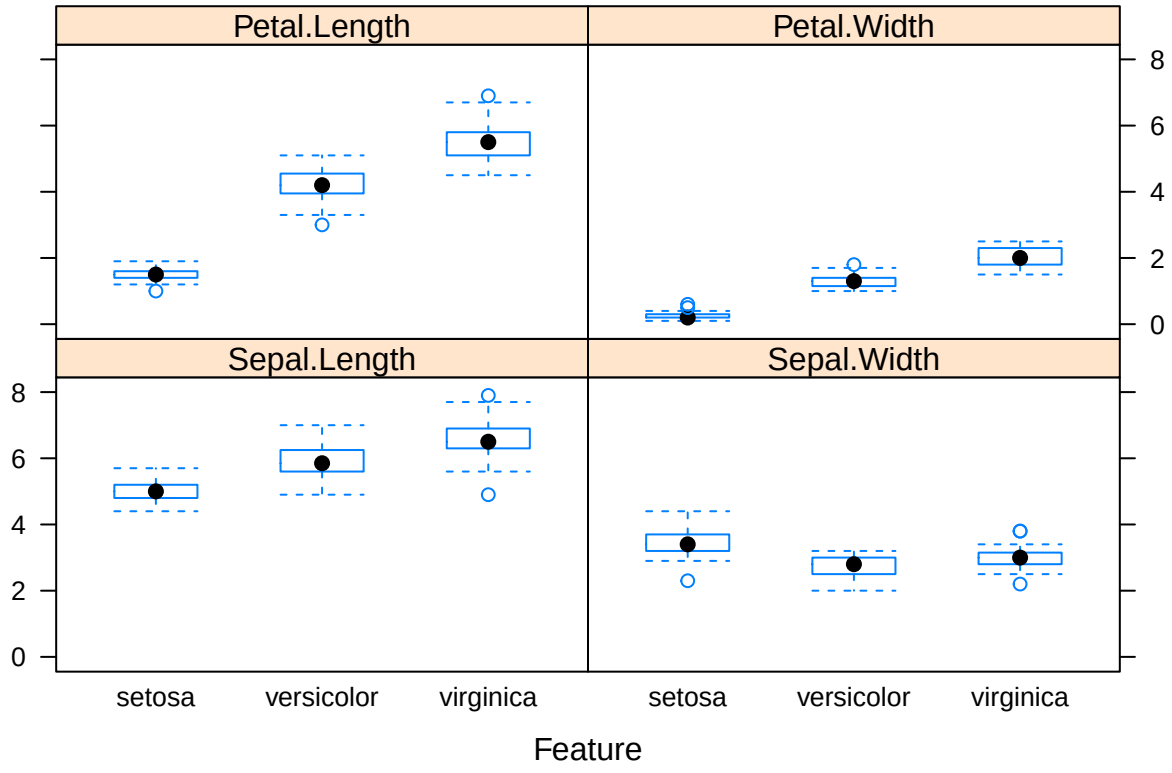


Scatter Plot Matrix

Multivariate boxplot

This boxplot shows the distribution of attributes per class level. There is a clear distinction per predictor and class. There are also some outlier values.

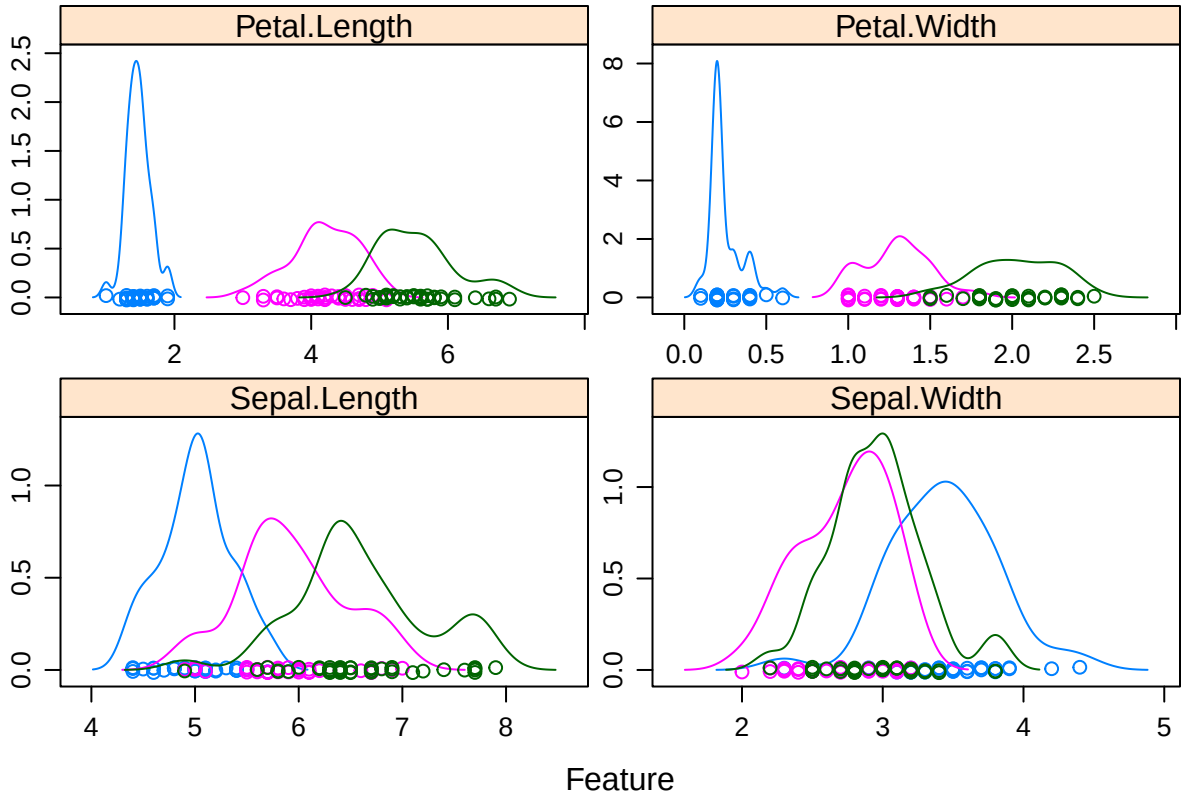
```
featurePlot(x = x, y = y, plot = "box")
```



Probability density plots

This plot shows the probability density functions for the different attributes per class level. This gives us an impression of the probability distribution.

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



Algorithm fitting and evaluation

This section will fit the algorithms to the training data. The algorithms employed are *linear discriminant analysis* (LDA), *classification and regression trees* (CART), *k-nearest neighbours* (kNN), *support vector machine* (SVM) and *random forest* (RF).

Training models

First, the control harness for the training of the models is defined, along with an evaluation metric. We use *accuracy* to evaluate model performance: the percentage of correct predictions relative to total training data.

```
# Test harness
control <- trainControl(method = "cv", number = 10)
metric <- "Accuracy"

# Linear Discriminant Analysis (LDA)
set.seed(813675)
fit.lda <- train(Species~., data = dataset, method = "lda", metric = metric, trControl = control)

# Classification and Regression Trees (CART)
set.seed(813675)
fit.cart <- train(Species~., data = dataset, method = "rpart", metric = metric, trControl = control)

# k-Nearest Neighbours (kNN)
```

```r
set.seed(813675)
fit.knn <- train(Species~., data = dataset, method = "knn", metric = metric, trControl = control)

# Support Vector Machine (SVM) with linear kernel
set.seed(813675)
fit.svm <- train(Species~., data = dataset, method = "svmRadial", metric = metric, trControl = control)

# Random Forest (RF)
set.seed(813675)
fit.rf <- train(Species~., data = dataset, method = "rf", metric = metric, trControl = control)
```

Model results

Next, we extract the results from the fitting process, coerce the results into a table and display it. Moreover, we visualise the results because there is a population of accuracy metrics due to the $n = 10$ cross validation.

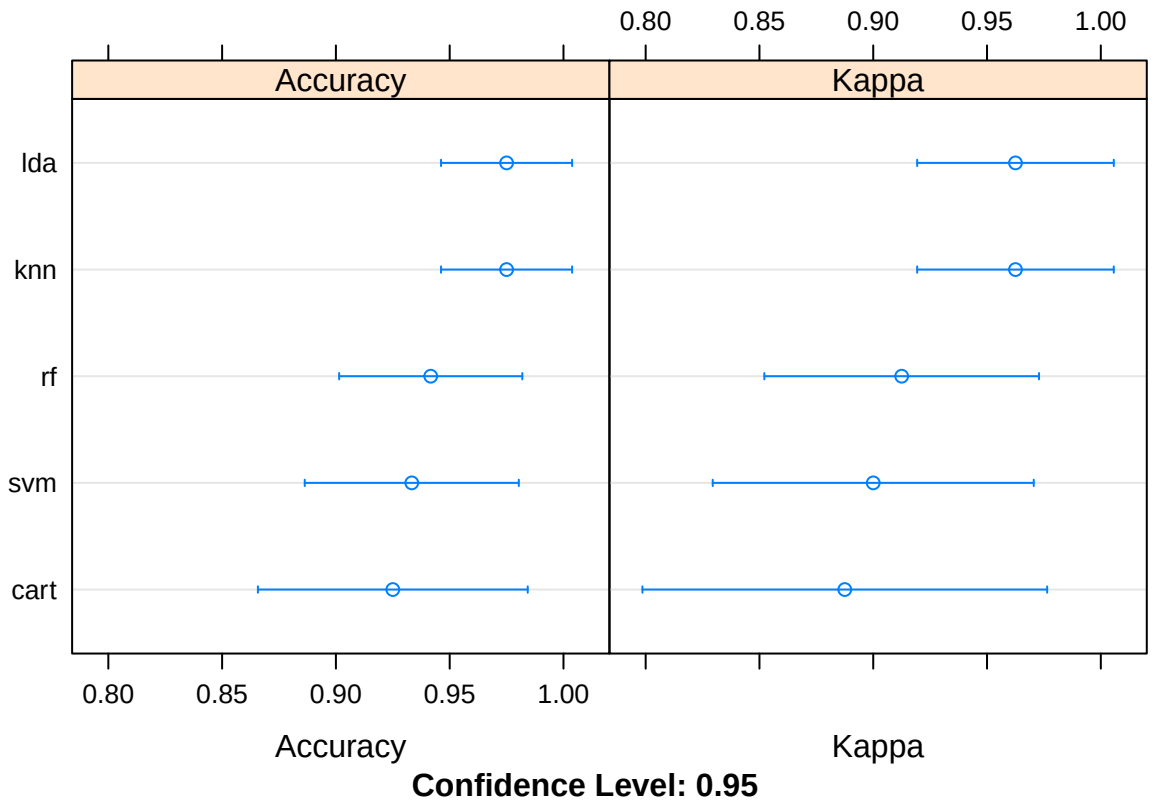
From both the table, as well as the visualisation, it can be seen that the **LDA algorithm** has yielded the best results; highest respective accuracy and kappa with the lowest relative volatility.

```r
## Summarise accuracy measures
results <- resamples(list(lda = fit.lda, cart = fit.cart, knn = fit.knn, svm = fit.svm, rf = fit.rf))

# Tabulate
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 0.9375000 1.0000000 0.9750000       1    1    0
## cart 0.7500000 0.9166667 0.9166667 0.9250000       1    1    0
## knn  0.9166667 0.9375000 1.0000000 0.9750000       1    1    0
## svm  0.8333333 0.9166667 0.9166667 0.9333333       1    1    0
## rf   0.8333333 0.9166667 0.9166667 0.9416667       1    1    0
##
## Kappa
##       Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
## lda  0.875 0.90625  1.000 0.9625       1    1    0
## cart 0.625 0.87500  0.875 0.8875       1    1    0
## knn  0.875 0.90625  1.000 0.9625       1    1    0
## svm  0.750 0.87500  0.875 0.9000       1    1    0
## rf   0.750 0.87500  0.875 0.9125       1    1    0
```

```r
# Visualise
dotplot(results)
```



Best fit

The specific results for the LDA fitted model are as follows:

```
# Print best algorithm's results
print(fit.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
```

Predictions and the test data

Subsequently, we want to make predictions using the best model for the particular data set; the LDA algorithm. To do so, the trained LDA model is applied to the test data set (validation set).

```
## Predictions
predictions <- predict(fit.lda, validation)
```

Confusion matrix

The results are now summarised in a confusion matrix. As can be seen in the tabulated results, the predictions in the test set have been *96.67%* accurate. The *Setosa* class has been predicted with perfect accuracy. One *versicolor* has been mistakenly predicted as *verginica*.

```
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
```