DECISION TREE: IRIS DATASET

Langat Erick

Decision trees are a popular machine learning technique for classification and regression analysis. They are particularly useful for data analysis because they can help *identify the most important variables in a dataset* and uncover patterns that may be hidden in the data.

Here's an example of how to conduct a decision tree analysis in R programming: First, we need to load the necessary libraries. In this example, we will be using the **rpart** library for decision trees:

```
library(rpart)
library(timetk)
library(tidymodels)
library(rpart.plot)
library(report)
```

Next, let's load a data-set to work with. In this example, we will be using the **iris** data-set, which is included in R. This data-set contains measurements for the sepal length, sepal width, petal length, and petal width of 150 iris flowers, along with their species (*setosa, versicolor, or virginica*):

```
data(iris)
colnames(iris)
```

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

Now, let's split the data-set into training and testing sets. We will use the training set to build our decision tree model, and the testing set to evaluate its accuracy:

```
set.seed(123)
# train_index <- sample(1:nrow(iris), nrow(iris)*0.7)
split <- initial_split(iris, prop = 0.7)
split
```

```
## <Training/Testing/Total>
## <105/45/150>
```

```
train_data <- training(split)
test_data <- testing(split)
```

Next, we will build our decision tree model using the **rpart()** function. In this example, we will be using the sepal length, sepal width, petal length, and petal width as predictor variables, and the iris species as the response variable:

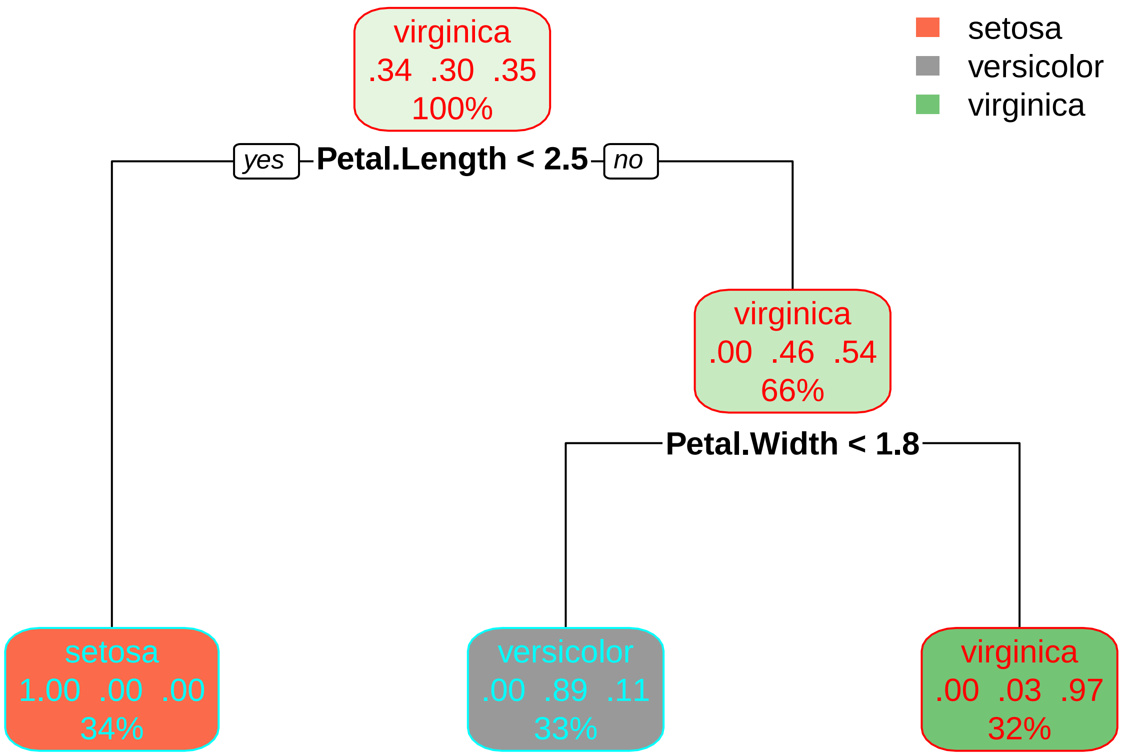
```
tree_model <- rpart(Species ~ ., data = train_data, method = "class")
summary(tree_model)
```

```
## Call:
## rpart(formula = Species ~ ., data = train_data, method = "class")
##   n= 105
##
##           CP nsplit  rel error    xerror       xstd
```

```
## 1 0.5294118      0 1.00000000 1.2058824 0.06232572
## 2 0.3970588      1 0.47058824 0.5441176 0.07198662
## 3 0.0100000      2 0.07352941 0.1176471 0.03997857
##
## Variable importance
##  Petal.Width Petal.Length Sepal.Length  Sepal.Width
##           34           32           21           13
##
## Node number 1: 105 observations,    complexity param=0.5294118
##   predicted class=virginica   expected loss=0.647619  P(node) =1
##     class counts:    36    32    37
##    probabilities: 0.343 0.305 0.352
##   left son=2 (36 obs) right son=3 (69 obs)
##   Primary splits:
##       Petal.Length < 2.45 to the left,  improve=35.54783, (0 missing)
##       Petal.Width  < 0.8  to the left,  improve=35.54783, (0 missing)
##       Sepal.Length < 5.45 to the left,  improve=24.79179, (0 missing)
##       Sepal.Width  < 3.25 to the right, improve=12.34670, (0 missing)
##   Surrogate splits:
##       Petal.Width  < 0.8  to the left,  agree=1.000, adj=1.000, (0 split)
##       Sepal.Length < 5.45 to the left,  agree=0.924, adj=0.778, (0 split)
##       Sepal.Width  < 3.25 to the right, agree=0.819, adj=0.472, (0 split)
##
## Node number 2: 36 observations
##   predicted class=setosa      expected loss=0  P(node) =0.3428571
##     class counts:    36     0     0
##    probabilities: 1.000 0.000 0.000
##
## Node number 3: 69 observations,    complexity param=0.3970588
##   predicted class=virginica   expected loss=0.4637681  P(node) =0.6571429
##     class counts:     0    32    37
##    probabilities: 0.000 0.464 0.536
##   left son=6 (35 obs) right son=7 (34 obs)
##   Primary splits:
##       Petal.Width  < 1.75 to the left,  improve=25.291950, (0 missing)
##       Petal.Length < 4.75 to the left,  improve=25.187810, (0 missing)
##       Sepal.Length < 6.15 to the left,  improve= 5.974246, (0 missing)
##       Sepal.Width  < 2.45 to the left,  improve= 2.411006, (0 missing)
##   Surrogate splits:
##       Petal.Length < 4.75 to the left,  agree=0.913, adj=0.824, (0 split)
##       Sepal.Length < 6.15 to the left,  agree=0.696, adj=0.382, (0 split)
##       Sepal.Width  < 2.65 to the left,  agree=0.638, adj=0.265, (0 split)
##
## Node number 6: 35 observations
##   predicted class=versicolor  expected loss=0.1142857  P(node) =0.3333333
##     class counts:     0    31     4
##    probabilities: 0.000 0.886 0.114
##
## Node number 7: 34 observations
##   predicted class=virginica   expected loss=0.02941176  P(node) =0.3238095
##     class counts:     0     1    33
##    probabilities: 0.000 0.029 0.971
```

We can visualize the decision tree using the **`plot()`** function:

```r
# plot(tree_model)
rpart.plot(tree_model,col=rainbow(2))
```



Finally, we can evaluate the accuracy of our decision tree model using the testing data:

```r
predicted_species <- predict(tree_model, test_data, type = "class")
predicted_species
```

```
##          1          2          3          5         11         18         19
##     setosa     setosa     setosa     setosa     setosa     setosa     setosa
##         28         29         33         36         45         48         49
##     setosa     setosa     setosa     setosa     setosa     setosa     setosa
##         55         56         57         58         59         61         62
## versicolor versicolor versicolor versicolor versicolor versicolor versicolor
##         65         66         68         70         77         83         84
## versicolor versicolor versicolor versicolor versicolor versicolor versicolor
##         94         95         98        100        101        104        105
## versicolor versicolor versicolor versicolor  virginica  virginica  virginica
##        111        113        116        125        131        133        135
##  virginica  virginica  virginica  virginica  virginica  virginica versicolor
##        140        141        145
##  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```r
accuracy <- sum(predicted_species == test_data$Species)/nrow(test_data)
accuracy
```

```
## [1] 0.9777778
```

This will give us the accuracy of our model on the testing data.

In summary, the above example demonstrates how to build a decision tree model using the `rpart` library in R programming. The decision tree helps us to identify the most important variables in a data-set and uncover patterns that may be hidden in the data. The model's accuracy can be evaluated using testing data, which allows us to see how well the model can generalize to new data.