Chapter 16: Introduction to Machine Learning

Ram Gopal, Dan Philps, and Tillman Weyde

2022

Contents

Load packages

```
library(caret)
library(rattle)
library(datasets)
library(rpart)
```

This chapter provides a nice introduction to machine learning through a decision tree example. Topics covered include preparing a dataset, training a model and evaluating its generalization, i.e. how it performs on new data that was not used for training.

Example 1: Fitting a Decision Tree for Classification

Read the iris dataset

```
data(iris)
print(unique(iris$Species))
```

```
## [1] setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

```
print(names(iris[,1:4]))
```

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

```
print(nrow(iris))
```

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

```
print(iris[1,])
```

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

Partition Data

- The caret package is a popular one to use for machine learning models in R. We are now going to divide the dataset into two parts, the training set for fitting the model (i.e. learn) and the test set that we use later to evaluate our model.

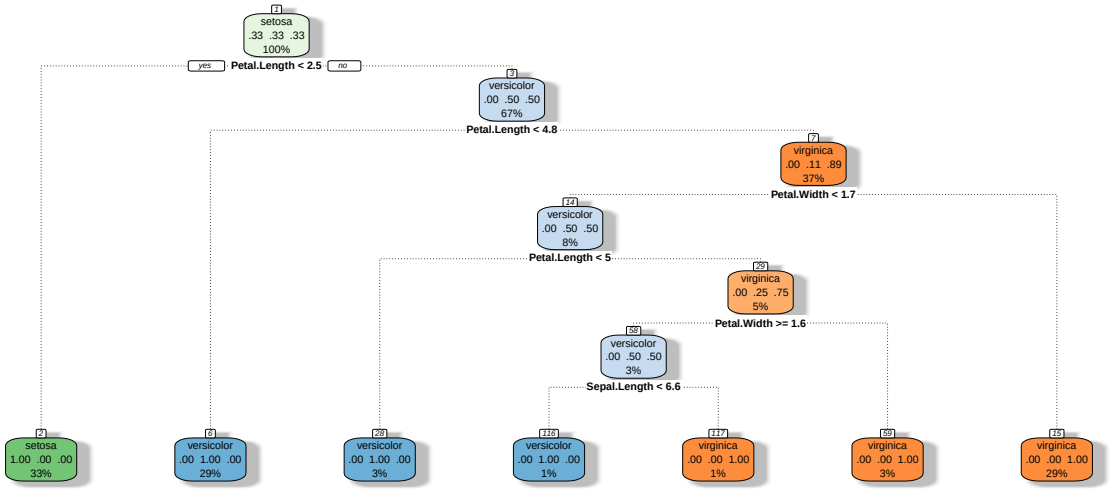
```
set.seed(123)
index <- createDataPartition(iris$Species,p=0.5,list=FALSE)
train <- iris[index,]
test <- iris[-index,]
```

Decision Tree

- For this chapter, we will use the rpart package for decision trees. In the subsequent chapters, we will employ the caret package that provides some uniformity and consistency across different machine learning algorithms. The function fancyRpartPlot from the rattle package creates fancy decision trees as seen below.

```
clf_full <- rpart(Species~., data=train,
                  control=rpart.control(minsplit=2, minbucket = 1, cp=-1))
fancyRpartPlot(clf_full,sub = '',main = "Full Decision Tree")
```

Full Decision Tree



Prediction and Accuracy

```
print(train[1,])
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 3          4.7         3.2          1.3         0.2  setosa
```

Training accuracy

- The predict function can be used to predict the outcome on new datasets. First, we evaluate how the model performs on the training data (which is the same data used for building the model).

```
prediction = predict(clf_full,newdata = train,type = "class")
accuracy = sum(prediction==train$Species)/nrow(train)
print(accuracy)
```

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

Accuracy on the test data

Now we can evaluate the model on a fresh, new dataset which in our case in the test data.Note that the test data has not been used at all in developing the decision tree. This allows us to evaluate how the model would perform in the future when we depoly the model.

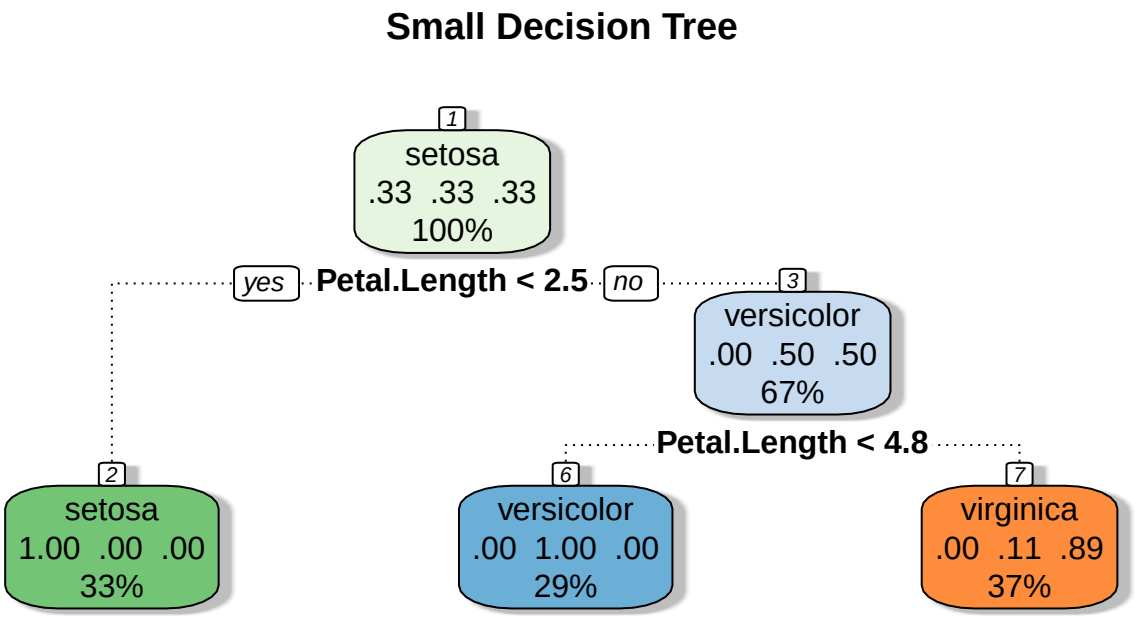
```
prediction = predict(clf_full,newdata = test,type = "class")
accuracy = sum(prediction==test$Species)/nrow(test)
print(accuracy)
```

```
## [1] 0.96
```

Regularizing the Model

Regularisation is a general term for techniques that prevent models from overfitting by limiting their ability to adapt. With a decision tree, a simple form of regularisation is to make the tree smaller. We will restrict the tree to a maximum depth of 2 below.

```
clf_small <- rpart(Species~., data=train,
                   control=rpart.control(maxdepth = 2))
fancyRpartPlot(clf_small,sub = '',main = "Small Decision Tree")
```



Calculate accuracy

When we calculate the test and training set accuracy for this simpler model we see that the values are much closer. This indicates that we have reduced overfitting and created a model that generalizes better.

```r
prediction = predict(clf_small,newdata = train,type = "class")
accuracy = sum(prediction==train$Species)/nrow(train)
print(paste("Training accuracy = ",accuracy))
```

```
## [1] "Training accuracy =  0.96"
```

```r
prediction = predict(clf_small,newdata = test,type = "class")
accuracy = sum(prediction==test$Species)/nrow(test)
print(paste("Test accuracy = ",accuracy))
```

```
## [1] "Test accuracy =  0.946666666666667"
```