title: “Chapter 3: Classification: Basic Concepts and Techniques” author: “Your Name” date: “2024-09-16” output: html\_document —

# Classification: Basic Concepts and Techniques

This chapter introduces decision trees for classification and discusses how models are built and evaluated. The corresponding chapter of the data mining textbook is available online: [Chapter 3: Classification: Basic Concepts and Techniques](link-to-textbook).

## Packages Used in this Chapter

pkgs <- c("caret", "FSelector", "lattice", "mlbench",   
 "palmerpenguins", "party", "pROC", "rpart", "rpart.plot",   
 "sampling", "tidyverse")  
pkgs\_install <- pkgs[!(pkgs %in% installed.packages()[,"Package"])]  
if(length(pkgs\_install)) install.packages(pkgs\_install)

The packages used for this chapter are:

* **caret** (M. Kuhn 2023)
* **FSelector** (Romanski Kotthoff and Schratz 2023)
* **lattice** (Sarkar 2023)
* **mlbench** (Leisch and Dimitriadou 2024)
* **palmerpenguins** (Horst Hill and Gorman 2022)
* **party** (Hothorn et al. 2024)
* **pROC** (Robin et al. 2023)
* **rpart** (Therneau and Atkinson 2023)
* **rpart.plot** (Milborrow 2024)
* **sampling** (Tillé and Matei 2023)
* **tidyverse** (Wickham 2023c)

A great cheat sheet for caret can be found [here](link-to-cheatsheet).

## Basic Concepts

Classification is a machine learning task where we learn a predictive function , where is the attribute set and is the class label. This is a **supervised learning** problem, where the class label is available in the training data.

A related supervised learning problem is **regression**, where is a number instead of a label.

## General Framework for Classification

Supervised learning has two steps:

1. **Induction**: Train a model on **training data** with known class labels.
2. **Deduction**: Predict class labels for new data.

We evaluate models on **test data** that does not overlap with the training data. The error on test data is called the **generalization error**.

## The Zoo Dataset

To demonstrate classification, we will use the Zoo dataset from the mlbench package.

data(Zoo, package = "mlbench")  
head(Zoo)

## hair feathers eggs milk airborne aquatic predator toothed backbone  
## aardvark TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE  
## antelope TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE  
## bass FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE TRUE  
## bear TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE  
## boar TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE  
## buffalo TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE  
## breathes venomous fins legs tail domestic catsize type  
## aardvark TRUE FALSE FALSE 4 FALSE FALSE TRUE mammal  
## antelope TRUE FALSE FALSE 4 TRUE FALSE TRUE mammal  
## bass FALSE FALSE TRUE 0 TRUE FALSE FALSE fish  
## bear TRUE FALSE FALSE 4 FALSE FALSE TRUE mammal  
## boar TRUE FALSE FALSE 4 TRUE FALSE TRUE mammal  
## buffalo TRUE FALSE FALSE 4 TRUE FALSE TRUE mammal

We convert the data frame to a tibble from tidyverse:

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

as\_tibble(Zoo, rownames = "animal")

## # A tibble: 101 × 18  
## animal hair feathers eggs milk airborne aquatic predator toothed backbone  
## <chr> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl>   
## 1 aardva… TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE   
## 2 antelo… TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE   
## 3 bass FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE TRUE   
## 4 bear TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE   
## 5 boar TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE   
## 6 buffalo TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE   
## 7 calf TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE   
## 8 carp FALSE FALSE TRUE FALSE FALSE TRUE FALSE TRUE TRUE   
## 9 catfish FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE TRUE   
## 10 cavy TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE   
## # ℹ 91 more rows  
## # ℹ 8 more variables: breathes <lgl>, venomous <lgl>, fins <lgl>, legs <int>,  
## # tail <lgl>, domestic <lgl>, catsize <lgl>, type <fct>

We will translate all TRUE/FALSE values into factors.

Zoo <- Zoo |>   
 mutate(across(where(is.logical),   
 ~ factor(.x, levels = c("TRUE", "FALSE")))) |>  
 mutate(across(where(is.character), factor))  
  
summary(Zoo)

## hair feathers eggs milk airborne aquatic predator   
## TRUE :43 TRUE :20 TRUE :59 TRUE :41 TRUE :24 TRUE :36 TRUE :56   
## FALSE:58 FALSE:81 FALSE:42 FALSE:60 FALSE:77 FALSE:65 FALSE:45   
##   
##   
##   
##   
##   
## toothed backbone breathes venomous fins legs   
## TRUE :61 TRUE :83 TRUE :80 TRUE : 8 TRUE :17 Min. :0.000   
## FALSE:40 FALSE:18 FALSE:21 FALSE:93 FALSE:84 1st Qu.:2.000   
## Median :4.000   
## Mean :2.842   
## 3rd Qu.:4.000   
## Max. :8.000   
##   
## tail domestic catsize type   
## TRUE :75 TRUE :13 TRUE :44 mammal :41   
## FALSE:26 FALSE:88 FALSE:57 bird :20   
## reptile : 5   
## fish :13   
## amphibian : 4   
## insect : 8   
## mollusc.et.al:10

## Decision Tree Classifiers

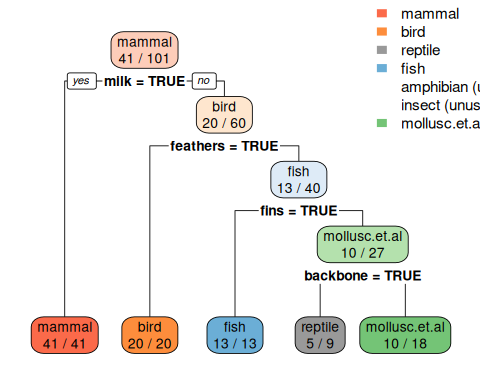
We’ll use the rpart package to build a decision tree.

library(rpart)  
tree\_default <- rpart(type ~ ., data = Zoo)  
tree\_default

## n= 101   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 101 60 mammal (0.41 0.2 0.05 0.13 0.04 0.079 0.099)   
## 2) milk=TRUE 41 0 mammal (1 0 0 0 0 0 0) \*  
## 3) milk=FALSE 60 40 bird (0 0.33 0.083 0.22 0.067 0.13 0.17)   
## 6) feathers=TRUE 20 0 bird (0 1 0 0 0 0 0) \*  
## 7) feathers=FALSE 40 27 fish (0 0 0.12 0.33 0.1 0.2 0.25)   
## 14) fins=TRUE 13 0 fish (0 0 0 1 0 0 0) \*  
## 15) fins=FALSE 27 17 mollusc.et.al (0 0 0.19 0 0.15 0.3 0.37)   
## 30) backbone=TRUE 9 4 reptile (0 0 0.56 0 0.44 0 0) \*  
## 31) backbone=FALSE 18 8 mollusc.et.al (0 0 0 0 0 0.44 0.56) \*

We can plot the resulting tree:

library(rpart.plot)  
rpart.plot(tree\_default, extra = 2)



### Predictions for New Data

Here’s how to make predictions for a new animal:

my\_animal <- tibble(hair = TRUE, feathers = TRUE, eggs = FALSE, milk = TRUE,   
 airborne = TRUE, aquatic = FALSE, predator = TRUE,  
 toothed = TRUE, backbone = TRUE, breathes = TRUE,  
 venomous = FALSE, fins = FALSE, legs = 4, tail = TRUE,   
 domestic = FALSE, catsize = FALSE, type = NA)  
  
my\_animal <- my\_animal |>   
 mutate(across(where(is.logical),   
 ~ factor(.x, levels = c("TRUE", "FALSE"))))  
  
predict(tree\_default, my\_animal, type = "class")

## 1   
## mammal   
## Levels: mammal bird reptile fish amphibian insect mollusc.et.al

## Model Evaluation

We calculate accuracy and error of the model on the training data.

pred <- predict(tree\_default, Zoo, type = "class")  
confusion\_table <- with(Zoo, table(type, pred))  
correct <- diag(confusion\_table) |> sum()  
error <- sum(confusion\_table) - correct  
accuracy <- correct / (correct + error)  
accuracy

## [1] 0.8811881

We can also use the caret package for evaluation.

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

confusionMatrix(data = pred, reference = Zoo$type)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction mammal bird reptile fish amphibian insect mollusc.et.al  
## mammal 41 0 0 0 0 0 0  
## bird 0 20 0 0 0 0 0  
## reptile 0 0 5 0 4 0 0  
## fish 0 0 0 13 0 0 0  
## amphibian 0 0 0 0 0 0 0  
## insect 0 0 0 0 0 0 0  
## mollusc.et.al 0 0 0 0 0 8 10  
##   
## Overall Statistics  
##   
## Accuracy : 0.8812   
## 95% CI : (0.8017, 0.9371)  
## No Information Rate : 0.4059   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8431   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: mammal Class: bird Class: reptile Class: fish  
## Sensitivity 1.0000 1.000 1.00000 1.0000  
## Specificity 1.0000 1.000 0.95833 1.0000  
## Pos Pred Value 1.0000 1.000 0.55556 1.0000  
## Neg Pred Value 1.0000 1.000 1.00000 1.0000  
## Prevalence 0.4059 0.198 0.04950 0.1287  
## Detection Rate 0.4059 0.198 0.04950 0.1287  
## Detection Prevalence 0.4059 0.198 0.08911 0.1287  
## Balanced Accuracy 1.0000 1.000 0.97917 1.0000  
## Class: amphibian Class: insect Class: mollusc.et.al  
## Sensitivity 0.0000 0.00000 1.00000  
## Specificity 1.0000 1.00000 0.91209  
## Pos Pred Value NaN NaN 0.55556  
## Neg Pred Value 0.9604 0.92079 1.00000  
## Prevalence 0.0396 0.07921 0.09901  
## Detection Rate 0.0000 0.00000 0.09901  
## Detection Prevalence 0.0000 0.00000 0.17822  
## Balanced Accuracy 0.5000 0.50000 0.95604

## Redoing the classification with iris dataset

Here’s the complete R code to perform the decision tree task using the **iris** dataset:

### Load Required Libraries

# Load necessary libraries  
library(rpart)  
library(rpart.plot)  
library(caret)

### 1. Load and Explore the Iris Dataset

# Load the iris dataset  
data(iris)  
  
# View the structure of the dataset  
str(iris)

## 'data.frame': 150 obs. of 5 variables:  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

# Display the first few rows of the dataset  
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

### 2. Build a Decision Tree

# Build the decision tree model with rpart  
tree\_iris <- rpart(Species ~ ., data = iris)  
  
# Print the tree structure  
print(tree\_iris)

## n= 150   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)   
## 2) Petal.Length< 2.45 50 0 setosa (1.00000000 0.00000000 0.00000000) \*  
## 3) Petal.Length>=2.45 100 50 versicolor (0.00000000 0.50000000 0.50000000)   
## 6) Petal.Width< 1.75 54 5 versicolor (0.00000000 0.90740741 0.09259259) \*  
## 7) Petal.Width>=1.75 46 1 virginica (0.00000000 0.02173913 0.97826087) \*

### 3. Plot the Decision Tree

# Plot the decision tree  
rpart.plot(tree\_iris, extra = 2)

### 4. Make Predictions

# Make predictions on the training data  
pred\_iris <- predict(tree\_iris, iris, type = "class")  
  
# View the first few predictions  
head(pred\_iris)

## 1 2 3 4 5 6   
## setosa setosa setosa setosa setosa setosa   
## Levels: setosa versicolor virginica

### 5. Evaluate the Model

# Calculate the confusion matrix and accuracy  
confusionMatrix(data = pred\_iris, reference = iris$Species)

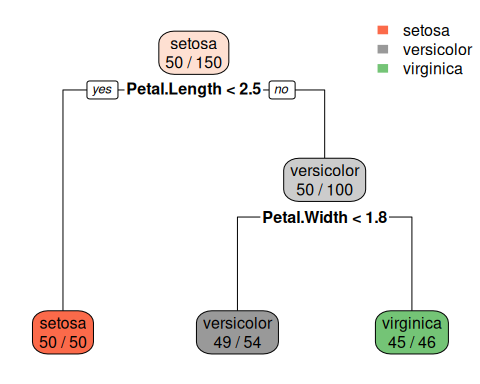
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction setosa versicolor virginica  
## setosa 50 0 0  
## versicolor 0 49 5  
## virginica 0 1 45  
##   
## Overall Statistics  
##   
## Accuracy : 0.96   
## 95% CI : (0.915, 0.9852)  
## No Information Rate : 0.3333   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.94   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: setosa Class: versicolor Class: virginica  
## Sensitivity 1.0000 0.9800 0.9000  
## Specificity 1.0000 0.9500 0.9900  
## Pos Pred Value 1.0000 0.9074 0.9783  
## Neg Pred Value 1.0000 0.9896 0.9519  
## Prevalence 0.3333 0.3333 0.3333  
## Detection Rate 0.3333 0.3267 0.3000  
## Detection Prevalence 0.3333 0.3600 0.3067  
## Balanced Accuracy 1.0000 0.9650 0.9450

### 6. Hyperparameter Tuning (Optional)

# Perform hyperparameter tuning using caret  
fit\_iris <- train(Species ~ ., data = iris, method = "rpart",   
 trControl = trainControl(method = "cv", number = 10),  
 tuneLength = 5)  
  
# Print the best model and accuracy  
print(fit\_iris)

## CART   
##   
## 150 samples  
## 4 predictor  
## 3 classes: 'setosa', 'versicolor', 'virginica'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa  
## 0.000 0.9333333 0.90   
## 0.125 0.9200000 0.88   
## 0.250 0.9200000 0.88   
## 0.375 0.9200000 0.88   
## 0.500 0.3333333 0.00   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.

# Plot the best tree from the tuning process  
rpart.plot(fit\_iris$finalModel, extra = 2)



### Explanation of Steps:

* **Loading the dataset**: The iris dataset is built into R, so it can be accessed directly.
* **Building the decision tree**: rpart is used to create a decision tree based on the features (Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) to predict the Species.
* **Plotting the tree**: rpart.plot is used to visualize the decision tree.
* **Making predictions**: The model predicts the species of flowers based on the tree.
* **Evaluating the model**: confusionMatrix from the caret package calculates the accuracy and provides a confusion matrix.
* **Hyperparameter tuning**: (Optional) The train function from caret performs cross-validation to find the best tree parameters.