Classification.R

Wow

Tue Oct 16 16:39:39 2018

```
############################################################
#                     Classification
############################################################
library(mlbench)
data(Zoo) #17 attributes, 101 animals, 7 classes = {Mammal, Bird, Reptile,
Fish, Amphibian, Insect, Mollusc.et.al.}
head(Zoo)
```

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

```
summary(Zoo)
```

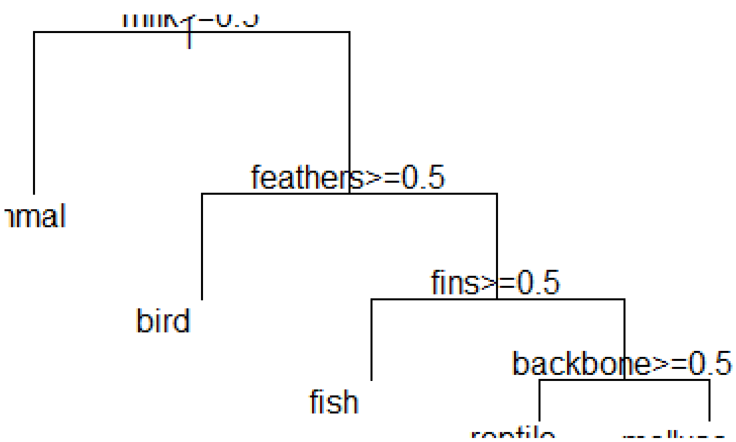
```
##     hair          feathers          eggs            milk
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:58        FALSE:81        FALSE:42        FALSE:60
##  TRUE :43        TRUE :20        TRUE :59        TRUE :41
##
##
##
##
##   airborne        aquatic         predator        toothed
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
```

```
##  FALSE:77        FALSE:65        FALSE:45        FALSE:40
##  TRUE :24        TRUE :36        TRUE :56        TRUE :61
##
##
##
##
##   backbone        breathes        venomous          fins
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:18        FALSE:21        FALSE:93        FALSE:84
##  TRUE :83        TRUE :80        TRUE :8         TRUE :17
##
##
##
##
##       legs            tail          domestic         catsize
##  Min.   :0.000   Mode :logical   Mode :logical   Mode :logical
##  1st Qu.:2.000   FALSE:26        FALSE:88        FALSE:57
##  Median :4.000   TRUE :75        TRUE :13        TRUE :44
##  Mean   :2.842
##  3rd Qu.:4.000
##  Max.   :8.000
##
##             type
##  mammal       :41
##  bird         :20
##  reptile      : 5
##  fish         :13
##  amphibian    : 4
##  insect       : 8
##  mollusc.et.al:10
```

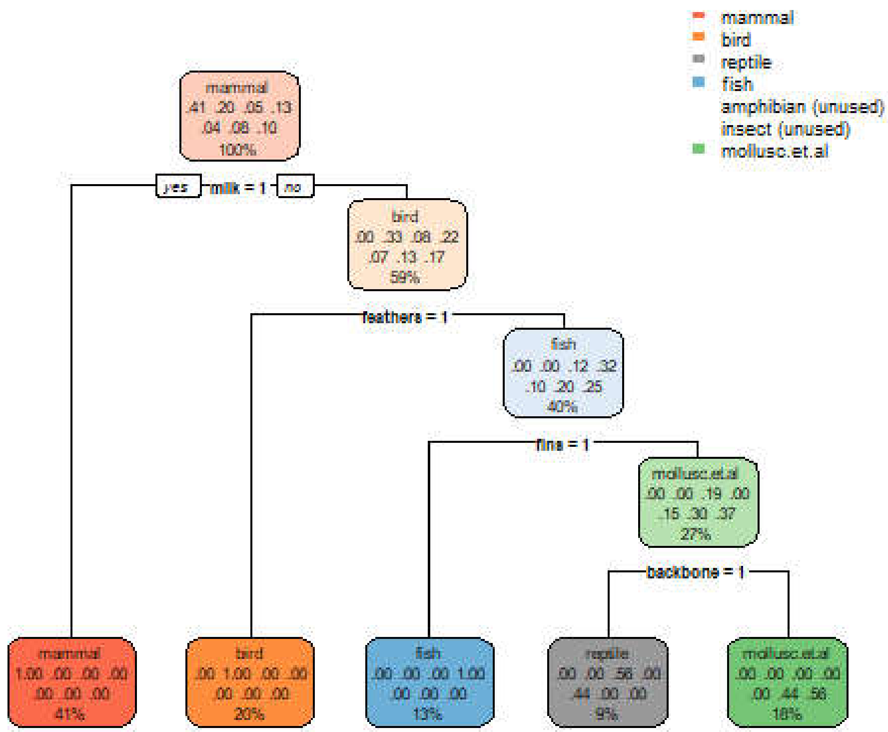
```
#############################################################
# Recursive Partitioning and Regression Trees (RPART)
# similar to Classification and regression trees (CART)
#############################################################
library(rpart)
tree1 <- rpart(type ~ ., data=Zoo) # predict "type" attribute using all the
remaining attributes
tree1 #TRUE =1, FALSE = 0
```

```
## 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>=0.5 41  0 mammal (1 0 0 0 0 0 0) *
##    3) milk< 0.5 60 40 bird (0 0.33 0.083 0.22 0.067 0.13 0.17)
##      6) feathers>=0.5 20  0 bird (0 1 0 0 0 0 0) *
##      7) feathers< 0.5 40 27 fish (0 0 0.12 0.32 0.1 0.2 0.25)
```

```
##       14) fins>=0.5 13  0 fish (0 0 0 1 0 0 0) *
##       15) fins< 0.5 27 17 mollusc.et.al (0 0 0.19 0 0.15 0.3 0.37)
##         30) backbone>=0.5 9  4 reptile (0 0 0.56 0 0.44 0 0) *
##         31) backbone< 0.5 18  8 mollusc.et.al (0 0 0 0 0 0.44 0.56) *

# meaning of the first line: 101 = # total animals at that node | 60 = #
misclassified animals | mammal = prediction result at that node | (0.41 0.2
0.05 0.13 0.04 0.079 0.099) = classes distribution (calculated from #total)
plot(tree1) #see ?plot.rpart
text(tree1)
```




```
# Better plotting
library(rpart.plot)
rpart.plot(tree1)
```




```
# Create a full tree (see: ?rpart.control)
tree2 <- rpart(type ~., data=Zoo, control=rpart.control(minsplit=2,cp=0.01))
# minsplit = the minimum number of observations that must exist in a node in
order for a split to be attempted.
# cp = complexity parameter. Any split that does not decrease the overall
lack of fit by a factor of cp is not attempted.
tree2

## 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>=0.5 41  0 mammal (1 0 0 0 0 0 0) *
##     3) milk< 0.5 60 40 bird (0 0.33 0.083 0.22 0.067 0.13 0.17)
##       6) feathers>=0.5 20  0 bird (0 1 0 0 0 0 0) *
##       7) feathers< 0.5 40 27 fish (0 0 0.12 0.32 0.1 0.2 0.25)
##        14) fins>=0.5 13  0 fish (0 0 0 1 0 0 0) *
##        15) fins< 0.5 27 17 mollusc.et.al (0 0 0.19 0 0.15 0.3 0.37)
##          30) backbone>=0.5 9  4 reptile (0 0 0.56 0 0.44 0 0)
##            60) aquatic< 0.5 4  0 reptile (0 0 1 0 0 0 0) *
##            61) aquatic>=0.5 5  1 amphibian (0 0 0.2 0 0.8 0 0)
##             122) eggs< 0.5 1  0 reptile (0 0 1 0 0 0 0) *
##             123) eggs>=0.5 4  0 amphibian (0 0 0 0 1 0 0) *
##          31) backbone< 0.5 18  8 mollusc.et.al (0 0 0 0 0 0.44 0.56)
##            62) airborne>=0.5 6  0 insect (0 0 0 0 0 1 0) *
```

```
##             63) airborne< 0.5 12  2 mollusc.et.al (0 0 0 0 0 0.17 0.83)
##              126) predator< 0.5 4  2 insect (0 0 0 0 0 0.5 0.5)
##                252) legs>=3 2  0 insect (0 0 0 0 0 1 0) *
##                253) legs< 3 2  0 mollusc.et.al (0 0 0 0 0 0 1) *
##              127) predator>=0.5 8  0 mollusc.et.al (0 0 0 0 0 0 1) *
```

```
rpart.plot(tree2)
```

```
## Warning: All boxes will be white (the box.palette argument will be
ignored) because
## the number of classes predicted by the model 7 is greater than
length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```

```
# Use the model to predict training records
# (See ?predict.rpart)
pred1 <- predict(tree1, Zoo) #return probability (as default)
pred1
```

```
##            mammal bird   reptile fish amphibian    insect mollusc.et.al
## aardvark        1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## antelope        1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## bass            0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## bear            1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## boar            1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## buffalo         1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## calf            1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## carp            0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## catfish         0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## cavy            1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## cheetah         1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## chicken         0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## chub            0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## clam            0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## crab            0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## crayfish        0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## crow            0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## deer            1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## dogfish         0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## dolphin         1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## dove            0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## duck            0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## elephant        1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## flamingo        0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## flea            0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## frog.1          0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## frog.2          0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## fruitbat        1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## giraffe         1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## girl            1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## gnat            0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
```

```
## goat          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## gorilla       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## gull          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## haddock       0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## hamster       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## hare          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## hawk          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## herring       0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## honeybee      0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## housefly      0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## kiwi          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## ladybird      0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## lark          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## leopard       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## lion          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## lobster       0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## lynx          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## mink          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## mole          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## mongoose      1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## moth          0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## newt          0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## octopus       0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## opossum       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## oryx          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## ostrich       0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## parakeet      0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## penguin       0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## pheasant      0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## pike          0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## piranha       0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## pitviper      0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## platypus      1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## polecat       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## pony          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## porpoise      1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## puma          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## pussycat      1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## raccoon       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## reindeer      1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## rhea          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## scorpion      0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## seahorse      0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## seal          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## sealion       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## seasnake      0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## seawasp       0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## skimmer       0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## skua          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## slowworm      0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
```

```
## slug          0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## sole          0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## sparrow       0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## squirrel      1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## starfish      0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## stingray      0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## swan          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## termite       0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## toad          0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## tortoise      0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## tuatara       0    0 0.5555556    0 0.4444444 0.0000000     0.0000000
## tuna          0    0 0.0000000    1 0.0000000 0.0000000     0.0000000
## vampire       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## vole          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## vulture       0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
## wallaby       1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## wasp          0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## wolf          1    0 0.0000000    0 0.0000000 0.0000000     0.0000000
## worm          0    0 0.0000000    0 0.0000000 0.4444444     0.5555556
## wren          0    1 0.0000000    0 0.0000000 0.0000000     0.0000000
```

```
pred2 <- predict(tree1, Zoo, type="class") #return a vector of classes
pred2
```

```
##      aardvark      antelope          bass          bear          boar
##        mammal        mammal          fish        mammal        mammal
##       buffalo          calf          carp       catfish          cavy
##        mammal        mammal          fish          fish        mammal
##       cheetah       chicken          chub          clam          crab
##        mammal          bird          fish mollusc.et.al mollusc.et.al
##      crayfish          crow          deer       dogfish       dolphin
## mollusc.et.al          bird        mammal          fish        mammal
##          dove          duck      elephant      flamingo          flea
##          bird          bird        mammal          bird mollusc.et.al
##        frog.1        frog.2      fruitbat       giraffe          girl
##       reptile       reptile        mammal        mammal        mammal
##          gnat          goat       gorilla          gull       haddock
## mollusc.et.al        mammal        mammal          bird          fish
##       hamster          hare          hawk       herring      honeybee
##        mammal        mammal          bird          fish mollusc.et.al
##      housefly          kiwi      ladybird          lark       leopard
## mollusc.et.al          bird mollusc.et.al          bird        mammal
##          lion       lobster          lynx          mink          mole
##        mammal mollusc.et.al        mammal        mammal        mammal
##      mongoose          moth          newt       octopus       opossum
##        mammal mollusc.et.al       reptile mollusc.et.al        mammal
##          oryx       ostrich      parakeet       penguin      pheasant
##        mammal          bird          bird          bird          bird
##          pike       piranha      pitviper      platypus       polecat
##          fish          fish       reptile        mammal        mammal
```

```
##          pony      porpoise          puma      pussycat       raccoon 
##        mammal        mammal        mammal        mammal        mammal 
##      reindeer          rhea      scorpion      seahorse          seal 
##        mammal          bird mollusc.et.al          fish        mammal 
##       sealion      seasnake       seawasp       skimmer          skua 
##        mammal       reptile mollusc.et.al          bird          bird 
##      slowworm          slug          sole       sparrow      squirrel 
##       reptile mollusc.et.al          fish          bird        mammal 
##      starfish      stingray          swan       termite          toad 
## mollusc.et.al          fish          bird mollusc.et.al       reptile 
##      tortoise       tuatara          tuna       vampire          vole 
##       reptile       reptile          fish        mammal        mammal 
##       vulture       wallaby          wasp          wolf          worm 
##          bird        mammal mollusc.et.al        mammal mollusc.et.al 
##          wren 
##          bird 
## Levels: mammal bird reptile fish amphibian insect mollusc.et.al
```

```
pred3 <- predict(tree1, Zoo, type="vector") #return a vector of numbers 
representing classes
pred3
```

```
## aardvark antelope     bass     bear     boar  buffalo     calf     carp 
##        1        1        4        1        1        1        1        4 
##  catfish     cavy  cheetah  chicken     chub     clam     crab crayfish 
##        4        1        1        2        4        7        7        7 
##     crow     deer  dogfish  dolphin     dove     duck elephant flamingo 
##        2        1        4        1        2        2        1        2 
##     flea   frog.1   frog.2 fruitbat  giraffe     girl     gnat     goat 
##        7        3        3        1        1        1        7        1 
##  gorilla     gull  haddock  hamster     hare     hawk  herring honeybee 
##        1        2        4        1        1        2        4        7 
## housefly     kiwi ladybird     lark  leopard     lion  lobster     lynx 
##        7        2        7        2        1        1        7        1 
##     mink     mole mongoose     moth     newt  octopus  opossum     oryx 
##        1        1        1        7        3        7        1        1 
##  ostrich parakeet  penguin pheasant     pike  piranha pitviper platypus 
##        2        2        2        2        4        4        3        1 
##  polecat     pony porpoise     puma pussycat  raccoon reindeer     rhea 
##        1        1        1        1        1        1        1        2 
## scorpion seahorse     seal  sealion seasnake  seawasp  skimmer     skua 
##        7        4        1        1        3        7        2        2 
## slowworm     slug     sole  sparrow squirrel starfish stingray     swan 
##        3        7        4        2        1        7        4        2 
##  termite     toad tortoise  tuatara     tuna  vampire     vole  vulture 
##        7        3        3        3        4        1        1        2 
##  wallaby     wasp     wolf     worm     wren 
##        1        7        1        7        2
```

```
# Create confusion matrix
confusion_table <- table(Zoo$type, pred2)
confusion_table

##                 pred2
##                  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         0      0             0
##   fish                0    0       0   13         0      0             0
##   amphibian           0    0       4    0         0      0             0
##   insect              0    0       0    0         0      0             8
##   mollusc.et.al       0    0       0    0         0      0            10

# Evaluate number of animals that were predicted correctly using tree1
correct <- sum(diag(confusion_table))
correct

## [1] 89

# Evaluate the training errors which is the number of misclassification
errors committed on training records.
error <- sum(confusion_table)-correct
error

## [1] 12

# Evaluate the accuracy
accuracy <- correct/(correct+error)
accuracy

## [1] 0.8811881

#############################################################
# Create function for accuracy
accuracy <- function(truth, prediction) { #2 inputs: a truth vector and a
prediction vector
  tbl <- table(truth, prediction) #create confusion matrix
  sum(diag(tbl))/sum(tbl) #calculate accuracy
}

# Apply "accuracy" function
accuracy(Zoo$type, pred2)

## [1] 0.8811881

# Training error of the full tree
accuracy(Zoo$type, predict(tree2, Zoo, type="class"))

## [1] 1
```

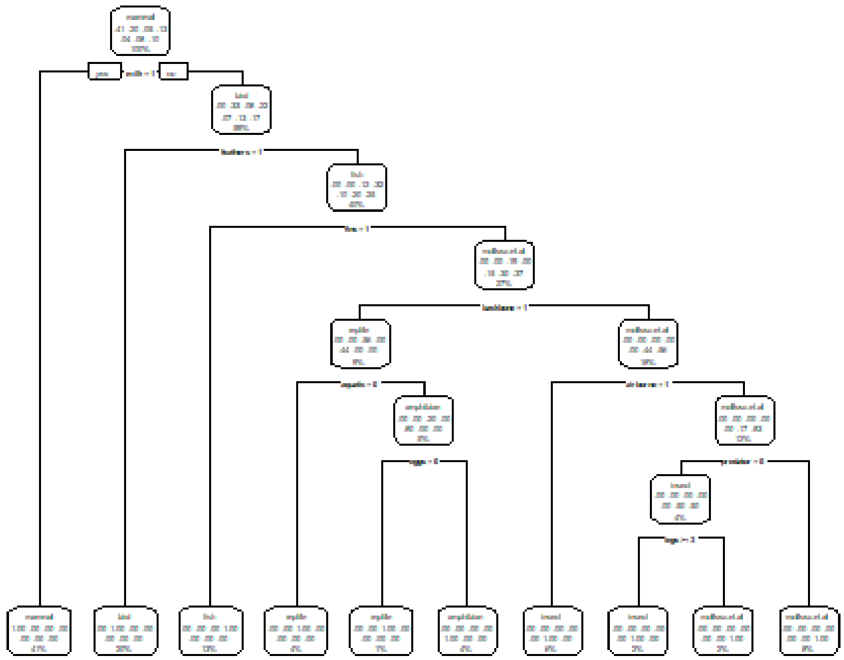
```
##################################################################
# Use rpart on a subset of attributes
tree3 <- rpart(type ~ hair+feathers+eggs+milk, data=Zoo) # predict "type"
attribute using only hair,feathers,eggs,mik attributes
tree3

## 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>=0.5 41  0 mammal (1 0 0 0 0 0 0) *
##   3) milk< 0.5 60 40 bird (0 0.33 0.083 0.22 0.067 0.13 0.17)
##     6) feathers>=0.5 20  0 bird (0 1 0 0 0 0 0) *
##     7) feathers< 0.5 40 27 fish (0 0 0.12 0.32 0.1 0.2 0.25) *

##################################################################
# Use rpart on a subset of records
tree4 <- rpart(type ~ ., data=Zoo, subset = c(1:50)) # predict "type"
attribute using only first 50 records
tree4

## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 27 mammal (0.46 0.18 0 0.14 0.04 0.1 0.08)
##   2) eggs< 0.5 23  0 mammal (1 0 0 0 0 0 0) *
##   3) eggs>=0.5 27 18 bird (0 0.33 0 0.26 0.074 0.19 0.15)
##     6) feathers>=0.5 9  0 bird (0 1 0 0 0 0 0) *
##     7) feathers< 0.5 18 11 fish (0 0 0 0.39 0.11 0.28 0.22) *

##################################################################
# Methods to evaluate the performance of a classifier
##################################################################
# 1. Hold-out: Create a decision tree using a training set and test a tree on
a test set
n_train <- as.integer(nrow(Zoo)*0.67) #2/3 of the records will be used as a
training set and 1/3 of the records will be used as a test set
train_id <- sample(1:nrow(Zoo), n_train)
train <- Zoo[train_id,]
test <- Zoo[-train_id, -17]
test_type <- Zoo[-train_id, 17]

# Create a decision tree
tree <- rpart(type ~., data=train,control=rpart.control(minsplit=2,cp=0.01))
tree
```

```
## n= 67
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 67 41 mammal (0.39 0.25 0.06 0.09 0.03 0.075 0.1)
##     2) milk>=0.5 26  0 mammal (1 0 0 0 0 0 0) *
##     3) milk< 0.5 41 24 bird (0 0.41 0.098 0.15 0.049 0.12 0.17)
##       6) feathers>=0.5 17  0 bird (0 1 0 0 0 0 0) *
##       7) feathers< 0.5 24 17 mollusc.et.al (0 0 0.17 0.25 0.083 0.21 0.29)
##        14) fins>=0.5 6  0 fish (0 0 0 1 0 0 0) *
##        15) fins< 0.5 18 11 mollusc.et.al (0 0 0.22 0 0.11 0.28 0.39)
##          30) tail>=0.5 4  0 reptile (0 0 1 0 0 0 0) *
##          31) tail< 0.5 14  7 mollusc.et.al (0 0 0 0 0.14 0.36 0.5)
##            62) breathes>=0.5 8  3 insect (0 0 0 0 0.25 0.62 0.12)
##             124) legs< 5 3  1 amphibian (0 0 0 0 0.67 0 0.33)
##               248) aquatic>=0.5 2  0 amphibian (0 0 0 0 1 0 0) *
##               249) aquatic< 0.5 1  0 mollusc.et.al (0 0 0 0 0 0 1) *
##             125) legs>=5 5  0 insect (0 0 0 0 0 1 0) *
##            63) breathes< 0.5 6  0 mollusc.et.al (0 0 0 0 0 0 1) *

# Training error
accuracy(train$type, predict(tree, train, type="class"))

## [1] 1

# Generalization error
accuracy(test_type, predict(tree, test, type="class"))

## [1] 0.9411765

#############################################################
# 2. Random Subsampling: It repeats holdout method several times.
k <- 10 #number of hold-out (could be changed)

# Do each hold-out
accs <- vector(mode="numeric") #create an empty numeric vector
for(i in 1:k){
  n_train <- as.integer(nrow(Zoo)*0.67)
  train_id <- sample(1:nrow(Zoo), n_train)
  train <- Zoo[train_id,]
  test <- Zoo[-train_id, -17]
  test_type <- Zoo[-train_id, 17]
  tree <- rpart(type ~.,
data=train,control=rpart.control(minsplit=2,cp=0.01))
  accs[i] <- accuracy(test_type, predict(tree, test, type="class"))
}
accs

##  [1] 0.9117647 0.9705882 1.0000000 0.9705882 0.9411765 0.9411765 0.9411765
##  [8] 0.9705882 0.9117647 0.9411765
```

```
# Report the average accuracy
mean(accs)

## [1] 0.95

##############################################################
# 3. Crossvalidation
# 3.1 k-fold cross-validation
k <- 10 #number of folds (could be changed)
index <- 1:nrow(Zoo)
index <- sample(index) # shuffle index
fold <- rep(1:k, each=nrow(Zoo)/k)[1:nrow(Zoo)] #make a repeat vector to be
the same size as index vector
folds <- split(index, fold) # split(x,f) = split divides the data in the
vector x into the groups defined by f.
folds

## $`1`
##  [1] 41 92 87 96 31 11 81 93 13 47
##
## $`2`
##  [1] 59 44 64 62 66 82 52 97 95 29
##
## $`3`
##  [1] 79  8 28  7 63 12 83 42 69 50
##
## $`4`
##  [1] 16 91 98  1 94 34 55 54 89 56
##
## $`5`
##  [1] 86 49 27 39 76 23 85 74 48 77
##
## $`6`
##  [1] 38 25  2 84 73  9 80 19 17 57
##
## $`7`
##  [1]  30  53   5   3  65 101   6  71  24  33
##
## $`8`
##  [1] 36 61 70 20 58 35 88 67 18 78
##
## $`9`
##  [1]  10  21 100  68  45  43  15  22  60  32
##
## $`10`
##  [1] 14 75 46 51 99 40 26 72  4 90

# Do each fold
accs <- vector(mode="numeric") #create an empty numeric vector
for(i in 1:length(folds)) {
  tree <- rpart(type ~., data=Zoo[-folds[[i]],],
```

```
control=rpart.control(minsplit=2,cp=0.01))
  accs[i] <- accuracy(Zoo[folds[[i]],]$type, predict(tree, Zoo[folds[[i]],],
type="class"))
}
accs

##  [1] 1.0 1.0 1.0 0.8 0.9 0.9 0.9 1.0 0.9 1.0

# Report the average accuracy
mean(accs)

## [1] 0.94

#############################################################
# 3.2 Leave-one-out
accs <- vector(mode="numeric") #create an empty numeric vector
for(i in 1:nrow(Zoo)) {
  tree <- rpart(type ~., data=Zoo[-i,],
control=rpart.control(minsplit=2,cp=0.01))
  accs[i] <- accuracy(Zoo[i,]$type, predict(tree, Zoo[i,], type="class"))
}
accs

##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1
##  [36] 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
##  [71] 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1

# Report the average accuracy
mean(accs)

## [1] 0.9306931

#############################################################
# 4. Bootstrap
k <- 10 #number of rounds (could be changed)
n_train <- as.integer(nrow(Zoo)*0.67) #2/3 of the records will be used as a
training set and 1/3 of the records will be used as a test set
train_id <- sample(1:nrow(Zoo), n_train)
test <- Zoo[-train_id, -17]
test_type <- Zoo[-train_id, 17]

# Do each round
e <- vector(mode="numeric") #create an empty numeric vector
accs <- vector(mode="numeric") #create an empty numeric vector
for(i in 1:k) {
  btrain_id <- sample(train_id, n_train, replace = TRUE) #Bootstrap sampling
for training set
  btrain <- Zoo[btrain_id,]
  tree <- rpart(type ~., data=btrain,
control=rpart.control(minsplit=2,cp=0.01))
```

```r
  accs[i] <- accuracy(btrain[,17], predict(tree, btrain[,-17], type="class"))
  e[i] <- accuracy(test_type, predict(tree, test, type="class"))
}
accs

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

e

##  [1] 0.9117647 0.9411765 0.9411765 0.9411765 0.9117647 0.9705882 0.9411765
##  [8] 0.9117647 0.9705882 0.9705882

accboot <- (1/k)*(sum(0.632*e+0.368*accs))
accboot

## [1] 0.9628235

#############################################################
# Use caret package for easier model building and evaluation
#############################################################
# See http://topepo.github.io/caret/index.html
#     https://cran.r-project.org/web/packages/caret/caret.pdf

# Use multi-core to make R faster
library(foreach)
library(iterators)
library(parallel)
library(doParallel)
registerDoParallel()

# Evaluation with caret (train normaly tries to tune cp for rpart). By
setting tuneLength 0 and tuneGrid fixed the value to 0.01, rpart is run with
no tuning
# See ?trainControl for method options
library(ggplot2)
```



```
library(lattice)
library(caret)
library(e1071)
# 10 folds crossvalidation
fit <- train(Zoo[,-17], Zoo[,17],
             method = "rpart",
             control=rpart.control(minsplit=2),
             tuneGrid=data.frame(cp=0.01),
             trControl = trainControl(method = "cv", number = 10), #10 is 
number of folds
             tuneLength=0)
# if you get the error message -> reinstall package caret
fit

## CART
##
## 101 samples
##  16 predictor
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect', 
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 92, 91, 92, 91, 91, 89, ...
## Resampling results:
##
##   Accuracy   Kappa
```

```
##   0.9283333  0.9044795
##
## Tuning parameter 'cp' was held constant at a value of 0.01
```

```
# plot tree
rpart.plot(fit$finalModel)
```

```
## Warning: All boxes will be white (the box.palette argument will be
ignored) because
## the number of classes predicted by the model 7 is greater than
length(box.palette) 6.
## To silence this warning use box.palette=0 or trace=-1.
```

```
# Leave-one-out
fit <- train(Zoo[,-17], Zoo[,17], method = "rpart",
             control=rpart.control(minsplit=2),
             tuneGrid=data.frame(cp=0.01),
             trControl = trainControl(method = "LOOCV"),
             tuneLength=0)
fit
```

```
## CART
##
## 101 samples
##  16 predictor
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 100, 100, 100, 100, 100, 100, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9306931  0.9086445
##
## Tuning parameter 'cp' was held constant at a value of 0.01
```

```
# Bootstrap
fit <- train(Zoo[,-17], Zoo[,17], method = "rpart",
             control=rpart.control(minsplit=2),
             tuneGrid=data.frame(cp=0.01),
             trControl = trainControl(method = "boot", number = 10), #10 is
number of resampling iterations
             tuneLength=0)
fit
```

```
## CART
##
## 101 samples
##  16 predictor
```

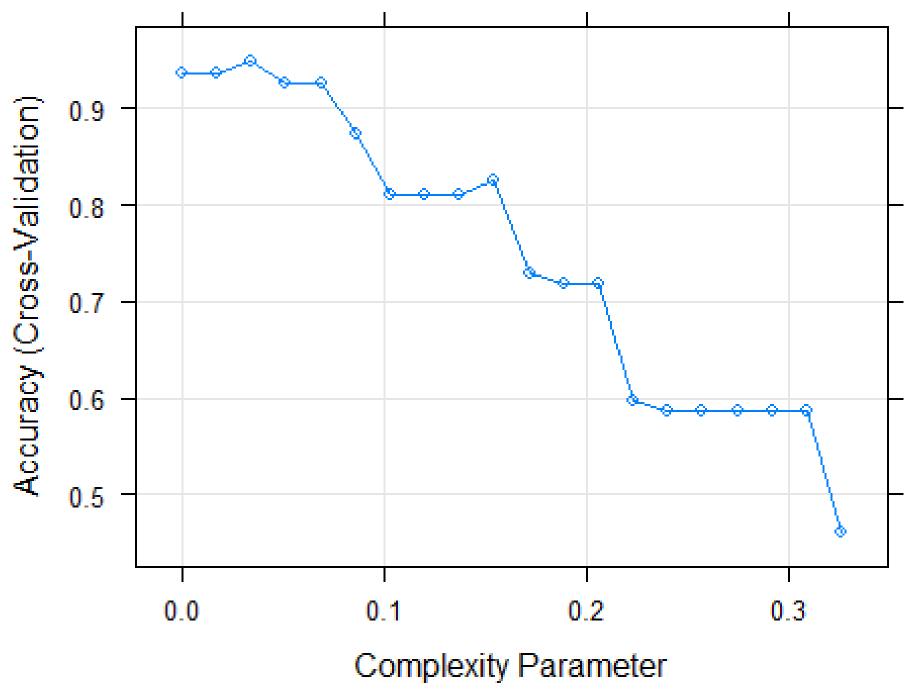
```
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 101, 101, 101, 101, 101, 101, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9145217  0.8879025
##
## Tuning parameter 'cp' was held constant at a value of 0.01
```

```
#############################################################
# Finding the best model
#############################################################
# Create training and test sets using createDataPartition function in caret
package
inTrain <- createDataPartition(y=Zoo$type, p = 0.75, list=FALSE)
training <- Zoo[inTrain,]
testing <- Zoo[-inTrain,]

# Find the best model
fit <- train(training[,-17], training[,17], method = "rpart",
             control=rpart.control(minsplit=2),
             trControl = trainControl(method = "cv", number = 10),
             tuneLength=20) #vary cp (complexity parameter) for 20 values and
pick the one that give the highest accuracy
fit
```

```
## CART
##
## 77 samples
## 16 predictors
##  7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 68, 71, 68, 69, 69, 69, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.9353175  0.9155613
##   0.01716247  0.9353175  0.9155613
##   0.03432494  0.9496032  0.9350058
##   0.05148741  0.9259921  0.9039251
##   0.06864989  0.9259921  0.9039251
##   0.08581236  0.8738095  0.8363539
##   0.10297483  0.8109127  0.7543430
```

```
##    0.12013730  0.8109127  0.7538175
##    0.13729977  0.8109127  0.7534773
##    0.15446224  0.8251984  0.7708750
##    0.17162471  0.7295635  0.6398750
##    0.18878719  0.7184524  0.6258125
##    0.20594966  0.7184524  0.6231458
##    0.22311213  0.5978175  0.4432473
##    0.24027460  0.5867063  0.4254348
##    0.25743707  0.5867063  0.4254348
##    0.27459954  0.5867063  0.4254348
##    0.29176201  0.5867063  0.4254348
##    0.30892449  0.5867063  0.4254348
##    0.32608696  0.4603175  0.1650000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03432494.
```

```
plot(fit)
```




```
# Use the best model
pred <- predict(fit, testing)
pred
```

```
##  [1] mammal         mammal         mammal         bird           mammal
##  [6] mammal         amphibian      mammal         mammal         bird
## [11] bird           mammal         mammal         bird           fish
## [16] mammal         insect         mollusc.et.al  bird           reptile
```

```
## [21] fish          fish          insect        insect
## Levels: mammal bird reptile fish amphibian insect mollusc.et.al
```

```r
# Confusion matrix (incl. confidence interval)
confusionMatrix(data = pred, testing$type)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        mammal bird reptile fish amphibian insect mollusc.et.al
##   mammal              10    0       0    0         0      0             0
##   bird                 0    5       0    0         0      0             0
##   reptile              0    0       0    0         0      0             1
##   fish                 0    0       0    3         0      0             0
##   amphibian            0    0       0    0         1      0             0
##   insect               0    0       0    0         0      2             1
##   mollusc.et.al        0    0       1    0         0      0             0
##
## Overall Statistics
##
##                Accuracy : 0.875
##                  95% CI : (0.6764, 0.9734)
##     No Information Rate : 0.4167
##     P-Value [Acc > NIR] : 4.596e-06
##
##                   Kappa : 0.8333
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: mammal Class: bird Class: reptile Class: fish
## Sensitivity                 1.0000      1.0000        0.00000       1.000
## Specificity                 1.0000      1.0000        0.95652       1.000
## Pos Pred Value              1.0000      1.0000        0.00000       1.000
## Neg Pred Value              1.0000      1.0000        0.95652       1.000
## Prevalence                  0.4167      0.2083        0.04167       0.125
## Detection Rate              0.4167      0.2083        0.00000       0.125
## Detection Prevalence        0.4167      0.2083        0.04167       0.125
## Balanced Accuracy           1.0000      1.0000        0.47826       1.000
##                      Class: amphibian Class: insect Class: mollusc.et.al
## Sensitivity                   1.00000       1.00000              0.00000
## Specificity                   1.00000       0.95455              0.95455
## Pos Pred Value                1.00000       0.66667              0.00000
## Neg Pred Value                1.00000       1.00000              0.91304
## Prevalence                    0.04167       0.08333              0.08333
## Detection Rate                0.04167       0.08333              0.00000
## Detection Prevalence          0.04167       0.12500              0.04167
## Balanced Accuracy             1.00000       0.97727              0.47727
```

```
###########################################################
# Other classification methods in caret package
###########################################################
# See http://topepo.github.io/caret/train-models-by-tag.html

# Create fixed sampling scheme (10-folds)
train <- createFolds(Zoo$type, k=10)

test <- Zoo[1:50,-17]
test_type <- Zoo[1:50,17]

#####################################################
# 1. Recursive Partitioning and Regression Trees (RPART)
# similar to Classification and regression trees (CART)
# Tuning parameters: cp (Complexity Parameter)

library(rpart)
rpartFit <- train(Zoo[,-17], Zoo[,17], "rpart",
                  tuneLength = 10,
                  trControl = trainControl(method = "cv", indexOut = train))
rpartFit

## CART
##
## 101 samples
##  16 predictor
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 89, 92, 92, 91, 91, 91, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.8965657  0.8661267
##   0.03703704  0.8965657  0.8661267
##   0.07407407  0.8683838  0.8300522
##   0.11111111  0.8378283  0.7884265
##   0.14814815  0.8378283  0.7884265
##   0.18518519  0.7377778  0.6502162
##   0.22222222  0.6103030  0.4563226
##   0.25925926  0.6103030  0.4563226
##   0.29629630  0.6103030  0.4563226
##   0.33333333  0.4690909  0.1404695
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03703704.

rpartFit$finalModel
```

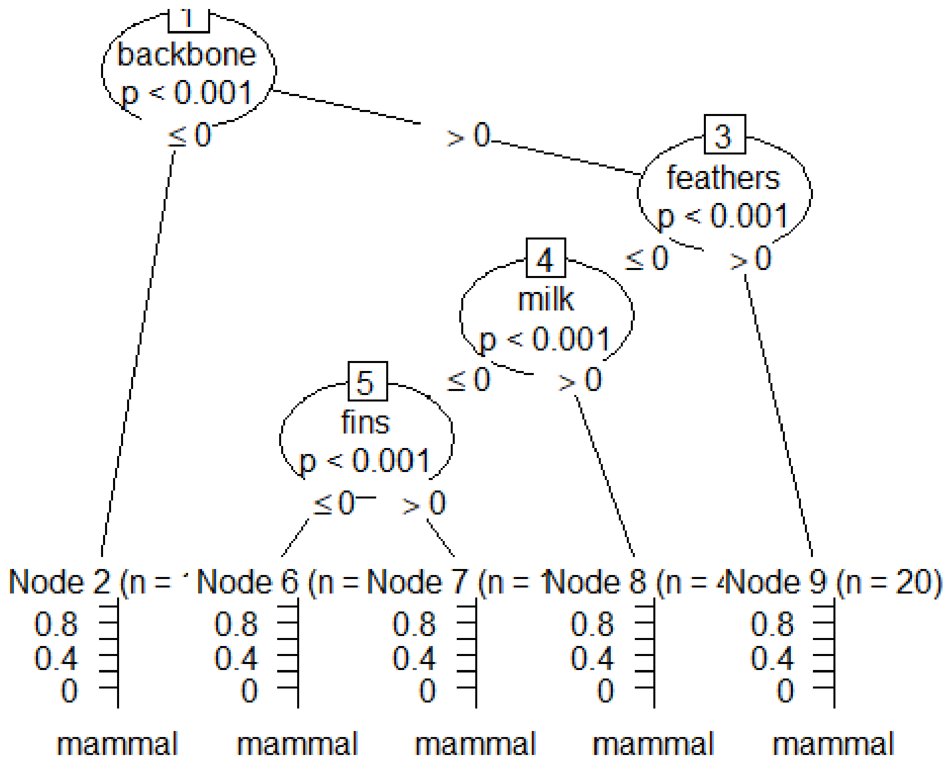
```
## n= 101
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 101 60 mammal (0.41 0.2 0.05 0.13 0.04 0.079 0.099)
##    2) milk>=0.5 41  0 mammal (1 0 0 0 0 0 0) *
##    3) milk< 0.5 60 40 bird (0 0.33 0.083 0.22 0.067 0.13 0.17)
##      6) feathers>=0.5 20  0 bird (0 1 0 0 0 0 0) *
##      7) feathers< 0.5 40 27 fish (0 0 0.12 0.32 0.1 0.2 0.25)
##       14) fins>=0.5 13  0 fish (0 0 0 1 0 0 0) *
##       15) fins< 0.5 27 17 mollusc.et.al (0 0 0.19 0 0.15 0.3 0.37)
##         30) backbone>=0.5 9  4 reptile (0 0 0.56 0 0.44 0 0) *
##         31) backbone< 0.5 18  8 mollusc.et.al (0 0 0 0 0 0.44 0.56) *

accs <- accuracy(test_type, predict(rpartFit, test))
accs

## [1] 0.86

#####################################################
# 2. Conditional Inference Tree (Decision Tree) (Ctree)
# Tuning parameters: mincriterion (1 - P-Value Threshold)

library(party)

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

ctreeFit <- train(Zoo[,-17], Zoo[,17], "ctree",
                  tuneLength = 10,
                  trControl = trainControl(method = "cv", indexOut = train))
ctreeFit
```

```
## Conditional Inference Tree
##
## 101 samples
##  16 predictor
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 90, 92, 91, 92, 90, 92, ...
## Resampling results across tuning parameters:
##
##   mincriterion  Accuracy   Kappa
##   0.0100000     0.8671212  0.8265623
##   0.1188889     0.8671212  0.8265623
##   0.2277778     0.8671212  0.8265623
##   0.3366667     0.8671212  0.8265623
##   0.4455556     0.8671212  0.8265623
##   0.5544444     0.8671212  0.8265623
##   0.6633333     0.8671212  0.8265623
##   0.7722222     0.8671212  0.8265623
##   0.8811111     0.8671212  0.8265623
##   0.9900000     0.8671212  0.8265623
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mincriterion = 0.99.
```

```
ctreeFit$finalModel
```

```
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  .outcome
## Inputs:  hair, feathers, eggs, milk, airborne, aquatic, predator, toothed,
backbone, breathes, venomous, fins, legs, tail, domestic, catsize
## Number of observations:  101
##
## 1) backbone <= 0; criterion = 1, statistic = 100
##   2)*  weights = 18
## 1) backbone > 0
##   3) feathers <= 0; criterion = 1, statistic = 82
##     4) milk <= 0; criterion = 1, statistic = 62
##       5) fins <= 0; criterion = 1, statistic = 21
##         6)*  weights = 9
##       5) fins > 0
##         7)*  weights = 13
##     4) milk > 0
##       8)*  weights = 41
##   3) feathers > 0
##     9)*  weights = 20
```

```r
plot(ctreeFit$finalModel)
```




```r
accs <- accuracy(test_type, predict(ctreeFit, test))
accs

## [1] 0.86

#####################################################
# 3. Support Vector Machines with Linear Kernel
# Tuning parameters: C (Cost)

library(e1071)
svmFit <- train(Zoo[,-17], Zoo[,17], "svmLinear2",
                tuneLength = 10,
                trControl = trainControl(method = "cv", indexOut = train))
svmFit

## Support Vector Machines with Linear Kernel
##
## 101 samples
##  16 predictor
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 90, 90, 93, 92, 91, 91, ...
```

```
## Resampling results across tuning parameters:
##
##   cost    Accuracy  Kappa
##     0.25  1         1
##     0.50  1         1
##     1.00  1         1
##     2.00  1         1
##     4.00  1         1
##     8.00  1         1
##    16.00  1         1
##    32.00  1         1
##    64.00  1         1
##   128.00  1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cost = 0.25.
```

```
svmFit$finalModel
```

```
##
## Call:
## svm.default(x = as.matrix(x), y = y, kernel = "linear", cost = param$cost,
##     probability = classProbs)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.25
##       gamma:  0.0625
##
## Number of Support Vectors:  48
```

```
accs <- accuracy(test_type, predict(svmFit, test))
accs
```

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

```
##############################################################
# 4. k-Nearest Neighbors
# Tuning parameters: k (neighbors)

knnFit <- train(Zoo[,-17], Zoo[,17], "knn",
                tuneLength = 10,
                trControl = trainControl(method = "cv", indexOut = train))
knnFit
```

```
## k-Nearest Neighbors
##
## 101 samples
##  16 predictor
```

```
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 92, 93, 91, 91, 90, 91, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9107071  0.8813892
##    7  0.8599495  0.8147876
##    9  0.8317677  0.7753775
##   11  0.7924747  0.7228499
##   13  0.7833838  0.7091086
##   15  0.7833838  0.7091086
##   17  0.7722727  0.6920650
##   19  0.7611616  0.6750215
##   21  0.7329798  0.6285396
##   23  0.7044444  0.5834563
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
knnFit$finalModel
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##        mammal          bird       reptile          fish     amphibian
##            41            20             5            13             4
##        insect mollusc.et.al
##             8            10
```

```
accs <- accuracy(test_type, predict(knnFit, test))
accs
```

```
## [1] 0.94
```

```
#############################################################
# 5. Neural Network
# Tuning parameters: 1) size (#Hidden Units)
#                    2) decay (Weight Decay)

library(nnet)
nnetFit <- train(Zoo[,-17], Zoo[,17], "nnet",
                 tuneLength = 5,
                 trControl = trainControl(method = "cv", indexOut = train))
```

```
## # weights:  79
## initial  value 211.309671
## iter  10 value 83.470198
```

```
## iter  20 value 37.891184
## iter  30 value 21.258269
## iter  40 value 14.201612
## iter  50 value 12.050633
## iter  60 value 10.679612
## iter  70 value 9.866870
## iter  80 value 9.795289
## iter  90 value 9.783621
## iter 100 value 9.782283
## final  value 9.782283
## stopped after 100 iterations
```

```
nnetFit
```

```
## Neural Network
##
## 101 samples
##  16 predictor
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 92, 90, 92, 91, 90, 91, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy   Kappa
##   1     0e+00  0.7734343  0.6720283
##   1     1e-04  0.8317172  0.7674380
##   1     1e-03  0.8819697  0.8374811
##   1     1e-02  0.8469192  0.7980851
##   1     1e-01  0.7195960  0.6222346
##   3     0e+00  0.8946970  0.8601876
##   3     1e-04  0.9718182  0.9621580
##   3     1e-03  1.0000000  1.0000000
##   3     1e-02  1.0000000  1.0000000
##   3     1e-01  0.9423737  0.9238240
##   5     0e+00  0.9888889  0.9844828
##   5     1e-04  1.0000000  1.0000000
##   5     1e-03  1.0000000  1.0000000
##   5     1e-02  1.0000000  1.0000000
##   5     1e-01  1.0000000  1.0000000
##   7     0e+00  1.0000000  1.0000000
##   7     1e-04  1.0000000  1.0000000
##   7     1e-03  1.0000000  1.0000000
##   7     1e-02  1.0000000  1.0000000
##   7     1e-01  1.0000000  1.0000000
##   9     0e+00  1.0000000  1.0000000
##   9     1e-04  1.0000000  1.0000000
##   9     1e-03  1.0000000  1.0000000
```

```
##   9     1e-02  1.0000000  1.0000000
##   9     1e-01  1.0000000  1.0000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.01.
```

```
nnetFit$finalModel
```

```
## a 16-3-7 network with 79 weights
## inputs: hairTRUE feathersTRUE eggsTRUE milkTRUE airborneTRUE aquaticTRUE
predatorTRUE toothedTRUE backboneTRUE breathesTRUE venomousTRUE finsTRUE legs
tailTRUE domesticTRUE catsizeTRUE
## output(s): .outcome
## options were - softmax modelling  decay=0.01
```

```
accs <- accuracy(test_type, predict(nnetFit, test))
accs
```

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

```
#####################################################
# 6. Random Forest
# Tuning parameters: mtry (#Randomly Selected Predictors)

library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
randomForestFit <- train(Zoo[,-17], Zoo[,17], "rf",
                         tuneLength = 10,
                         trControl = trainControl(method = "cv", indexOut =
train))
randomForestFit
```

```
## Random Forest
##
## 101 samples
##  16 predictor
##   7 classes: 'mammal', 'bird', 'reptile', 'fish', 'amphibian', 'insect',
'mollusc.et.al'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
```

```
## Summary of sample sizes: 92, 92, 91, 92, 91, 90, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    1         1
##    3    1         1
##    5    1         1
##    6    1         1
##    8    1         1
##    9    1         1
##   11    1         1
##   12    1         1
##   14    1         1
##   16    1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
randomForestFit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.96%
## Confusion matrix:
##              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    1       2    1         1      0             0
## fish              0    0       0   13         0      0             0
## amphibian         0    0       0    0         4      0             0
## insect            0    0       0    0         0      8             0
## mollusc.et.al     0    0       0    0         0      1             9
##              class.error
## mammal               0.0
## bird                 0.0
## reptile              0.6
## fish                 0.0
## amphibian            0.0
## insect               0.0
## mollusc.et.al        0.1
```

```
accs <- accuracy(test_type, predict(randomForestFit, test))
accs
```

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

```
###########################################################
```