More on Decision Trees Classification Trees, Regression Trees & Naïve Bayes

Thilanka Munasinghe

Data Analytics

ITWS-4600/ITWS-6600/MATP-4450/CSCI-4960

Group 2 Lab 3, February 20, 2020

Regression Tree example

- We will be using the msleep dataset that comes under the ggplot2 packsge.
- You need to install the following library packages first

library(rpart)

library(rpart.plot)

library(rpart)
library(rpart.plot)
data("msleep")
str(msleep)
help("msleep") # read the documentation for the msleep dataset.it is about mammals sleep dataset
# observe the structure of the #msleep dataset
str(data)

```
library(rpart)
library(rpart.plot)
data("msleep")
str(msleep)
help("msleep") # read the documentation for the msleep dataset.
# It is about mammals sleep dataset
# observe the structure of the #msleep dataset
str(data)
```

creating a new data frame with the following columns included.

mSleepDF1 <- msleep[,c(3,6,10,11)] # 3 = vore ,6=sleep_total, 10=brainwt, 11=bodywt

observe the structure of the mSleepDF

str(mSleepDF1)

head(mSleepDF1)

```
# creating a new data frame with the following columns included.
mSleepDF1 <- msleep[,c(3,6,10,11)] # 3 = vore ,6=sleep_total, 10=brainwt, 11=bodywt
# observe the structure of the mSleepDF
str(mSleepDF1)
head(mSleepDF1)
```

Building the model

Building Regression Decision Tree that #predicts the total sleeping

hours of the mamals based on the other #variables available on the dataset

help("rpart") # Read the documentation for the rpart() function.

sleepModel_1 <- rpart(sleep_total ~ ., data=mSleepDF1, method = "anova")

method we are using here is anova becuase our target here is sleep_total is a numerical one.

sleepModel_1

```
# Building Regression Decision Tree that #predicts the total sleeping
# hours of the mamals based on the other #variables available on the dataset
help("rpart") # Read the documentation for the rpart() function.
sleepModel_1 <- rpart(sleep_total ~ ., data=mSleepDF1, method = "anova")
# method we are using here is anova becuase our target here is sleep_total is a numerical one.
sleepModel_1
```

Interpretation

```
> sleepModel_1
n= 83

node), split, n, deviance, yval
      * denotes terminal node

 1) root 83 1624.066000 10.433730
   2) bodywt>=167.947 9    7.868889  3.488889 *
   3) bodywt< 167.947 74 1129.325000 11.278380
     6) bodywt>=1.85 31  458.593500  9.361290
      12) vore=herbi 7   88.337140  6.642857 *
      13) vore=carni,insecti,omni 24  303.439600 10.154170
        26) brainwt>=0.136 13  128.669200  9.392308 *
        27) brainwt< 0.136 11  158.307300 11.054550 *
     7) bodywt< 1.85 43  474.662800 12.660470
      14) vore=omni 13  141.370800 11.638460 *
      15) vore=carni,herbi,insecti 30  313.829700 13.103330 *
```

Root node has 83 samples

Branches

Leaf Node represented with *

We started with 83 observation that is why (n = 83)

let's visualize this using rpart.plot()

help("rpart.plot")

rpart.plot(sleepModel_1, type = 3, fallen.leaves = TRUE)

type = 3, Draw separate split labels for the left and right directions.See the documentation

#fallen.leaves = TRUE, Default TRUE to position the leaf nodes at the bottom of the graph.

#It can be helpful to use FALSE if the graph is too crowded and the text size is too small.

rpart.plot(sleepModel_1, type = 3,digits = 3, fallen.leaves = TRUE) # with 3 digits

rpart.plot(sleepModel_1, type = 3,digits = 4, fallen.leaves = TRUE)

```
# let's visualize this using rpart.plot()
help("rpart.plot")
rpart.plot(sleepModel_1, type = 3, fallen.leaves = TRUE)
# type = 3, Draw separate split labels for the left and right directions.See the documentation
#fallen.leaves = TRUE,  Default TRUE to position the leaf nodes at the bottom of the graph.
#It can be helpful to use FALSE if the graph is too crowded and the text size is too small.
rpart.plot(sleepModel_1, type = 3,digits = 3, fallen.leaves = TRUE) # with 3 digits
rpart.plot(sleepModel_1, type = 3,digits = 4, fallen.leaves = TRUE)
```

```
library(rpart)
library(rpart.plot)
data("msleep")
str(msleep)
help("msleep") # read the documentation for the msleep dataset.
# It is about mammal’s sleep dataset
# observe the structure of the #msleep dataset
str(data)
# creating a new data frame with the following columns included.
mSleepDF1 <- msleep[,c(3,6,10,11)] # 3 = vore ,6=sleep_total, 10=brainwt, 11=bodywt
# observe the structure of the mSleepDF
str(mSleepDF1)
head(mSleepDF1)
# Building Regression Decision Tree that #predicts the total sleeping
# hours of the mamals based on the other #variables available on the dataset
help("rpart") # Read the documentation for the rpart() function.
sleepModel_1 <- rpart(sleep_total ~ ., data=mSleepDF1, method = "anova")
# method we are using here is anova becuase our target here is sleep_total is a numerical one.
sleepModel_1
# let's visualize this using rpart.plot()
help("rpart.plot")
rpart.plot(sleepModel_1, type = 3, fallen.leaves = TRUE)
# type = 3, Draw separate split labels for the left and right directions.See the documentation
#fallen.leaves = TRUE,  Default TRUE to position the leaf nodes at the bottom of the graph.
#It can be helpful to use FALSE if the graph is too crowded and the text size is too small.
rpart.plot(sleepModel_1, type = 3,digits = 3, fallen.leaves = TRUE) # with 3 digits
rpart.plot(sleepModel_1, type = 3,digits = 4, fallen.leaves = TRUE)
```

Ctree()

- We will be building the classification tree using ctree() function.
- Again, Iris dataset ! ☺

```
# instrall the C50 package
install.packages("C50")
require(C50)
# we will be using the iris dataset to do a #classfication
data("iris")
head(iris)
str(iris)
table(iris$Species)
```

```
> data("iris")
> head(iris) # head of the iris dataset
  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
> str(iris) # look at the structure of the dataset using str()
'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 ...
```

It is important to set the seed in order to get the same randomly generated numbers when we run the model over and over.

```
# set the seed
set.seed(9850)
# generate random numbers
grn <-runif(nrow(iris))
```


```
# set the seed
set.seed(9850)
# generate random numbers
grn <-runif(nrow(iris))
```

creating a randomized iris dataset , shuffling the dataset

we use the order() function along with the #random numbers we generated.

irisrand <-iris[order(grn),]

```
# creating a randomized iris dataset ,  shuffling the dataset
# we use the order() function along with the #random numbers we generated.
irisrand <-iris[order(grn),]
```

```
# obsrve that rows are now randomly shuffled.
str(irisrand)
classificationmodel1 <-C5.0(irisrand[1:100,-5], irisrand[1:100,5])
classificationmodel1
summary(classificationmodel1)
```

summary(classificationmodel1)

```
Read 100 cases (5 attributes) from undefined.data

Decision tree:

Petal.Length <= 1.9: setosa (34)
Petal.Length > 1.9:
:...Petal.Width > 1.6: virginica (29)
    Petal.Width <= 1.6:
    :...Petal.Length <= 4.9: versicolor (35)
        Petal.Length > 4.9: virginica (2)


Evaluation on training data (100 cases):

            Decision Tree
          ----------------
          Size      Errors

             4    0( 0.0%)   <<


           (a)   (b)   (c)    <-classified as
          ----  ----  ----
            34                (a): class setosa
                  35          (b): class versicolor
                        31    (c): class virginica
```


```
Evaluation on training data (100 cases):

            Decision Tree
          ----------------
          Size      Errors

             4    0( 0.0%)   <<


           (a)   (b)   (c)    <-classified as
          ----  ----  ----
            34                (a): class setosa
                  35          (b): class versicolor
                        31    (c): class virginica
```

now we will do the prediction using the #predict() function

We are using the remaining last 50 rows for #here starting from 101 row to 150th row

prediction1 <- predict(classificationmodel1,irisrand[101:150,])

prediction1

```
# now we will do the prediction using the predict() function
# We are using the remaining last 50 rows for here starting from 101 row to 150th row

prediction1 <- predict(classificationmodel1,irisrand[101:150,])
prediction1
```

we will use the confusion matrix to #understand our prediction

Read the documentation for the table() function in RStudio help

table(irisrand[101:150,5],prediction1)

you can write the same above line by defining what is the "predicted"

table(irisrand[101:150,5],Predicted = prediction1)

```
# we will use the confusion matricx to understand our prediction
# Read the documentation for the table() function in RStudio help
table(irisrand[101:150,5],prediction1)
# you can write the same above line by defining what is the "predicted"
##  table(irisrand[101:150,5],Predicted = prediction1)
```

we will use the confusion matrix to #understand our prediction

Read the documentation for the table() #function in RStudio help

table(irisrand[101:150,5],prediction1)

```
> table(irisrand[101:150,5],prediction1)
            prediction1
             setosa versicolor virginica
  setosa         16          0         0
  versicolor      0         12         3
  virginica       0          0        19
```

<u>Correctly predicted</u>

16 of them Setosa

12 of them Versicolor

19 of them Verginica

Model predicted 47/50 correctly, however, there were <u>3 of them Incorrectly predicted</u>.

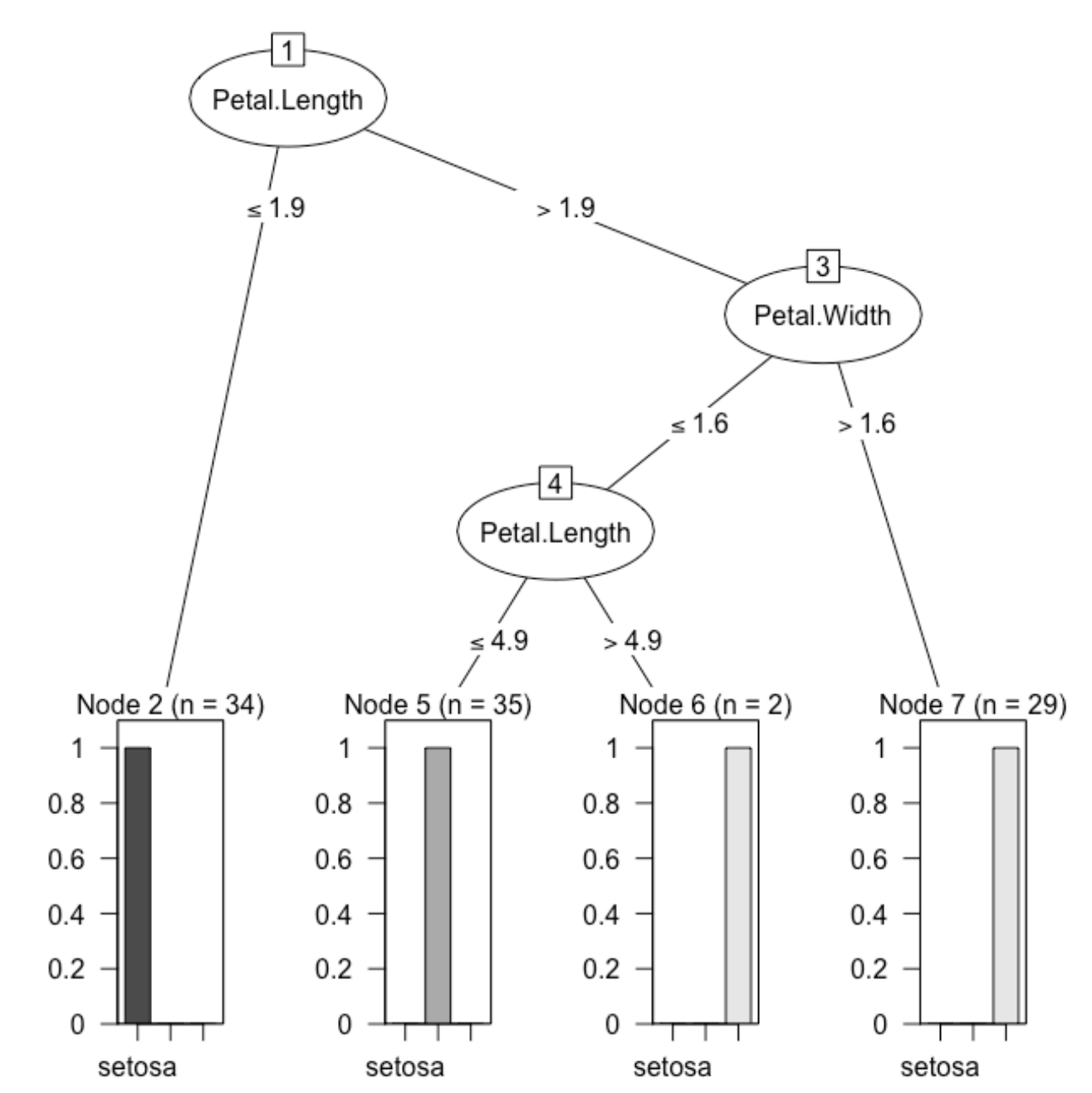
(There were 3 versicolor species out of the 50 examples that the model incorrectly classified as virginica)

Plotting the ctree

We can plot the classification model tree
#using the plot() function

plot(classificationmodel1)

```
# We can plot the classification model tree using the plot() function
plot(classificationmodel1)
```


1
Petal.Length
≤ 1.9
> 1.9
3
Petal.Width
≤ 1.6
> 1.6
4
Petal.Length
≤ 4.9
> 4.9
Node 2 (n = 34)
Node 5 (n = 35)
Node 6 (n = 2)
Node 7 (n = 29)
1
0.8
0.6
0.4
0.2
0
setosa

```
# install the C50 package
install.packages("C50")
# require(C50)
library(C50)
# we will be using the iris dataset to do a #classfication
data("iris")
head(iris) # head of the iris dataset
str(iris) # look at the structure of the dataset using str()
table(iris$Species) # using table() function we can look at the Species of Iris dataset column

# set the seed
set.seed(9850)
# generate random numbers
grn <-runif(nrow(iris))

# creating a randomized iris dataset ,  shuffling the dataset
# we use the order() function along with the #random numbers we generated.
irisrand <-iris[order(grn),]
# obsrve that rows are now randomly shuffled.
str(irisrand)
classificationmodel1 <-C5.0(irisrand[1:100,-5], irisrand[1:100,5])
classificationmodel1
summary(classificationmodel1)

# now we will do the prediction using the #predict() function
# We are using the remaining last 50 rows for #here starting from 101 row to 150th row
prediction1 <- predict(classificationmodel1,irisrand[101:150,])
prediction1

# we will use the confusion matrix to #understand our prediction
# Read the documentation for the table() function in RStudio help
table(irisrand[101:150,5],prediction1)
# you can write the same above line by defining what is the "predicted"
##  table(irisrand[101:150,5],Predicted = prediction1)
```

rpart() and ctree()

- Now you have used both rpart() and ctree() to generate the decision tree models.

- Make sure to read the documentation of ctree()

https://www.rdocumentation.org/packages/partykit/versions/1.2-3/topics/ctree

- Also go over the NaiveBayes Classifier example I shared in the lecture (code is in the next slide)
- Read the documentation of the Packages e1071 and C50 during your spare time, it will be very handy/useful down the road

Documentation of the Package e1071

https://cran.r-project.org/web/packages/e1071/e1071.pdf

Documentation of the Package C50

https://cran.r-project.org/web/packages/C50/C50.pdf

Digging into iris

```
## Call the NaiveBayes Classifier Package e1071, which auto calls the Class package ##
library("e1071")
classifier<-naiveBayes(iris[,1:4], iris[,5])
table(predict(classifier, iris[,-5]), iris[,5], dnn=list('predicted','actual'))
classifier$apriori
classifier$tables$Petal.Length
plot(function(x) dnorm(x, 1.462, 0.1736640), 0, 8, col="red", main="Petal length distribution for the 3 different species")
curve(dnorm(x, 4.260, 0.4699110), add=TRUE, col="blue")
curve(dnorm(x, 5.552, 0.5518947 ), add=TRUE, col = "green")
```

mean

Standard Deviation

- Now, go over and explore on **remaining** R scripts that are given in the website (group2: https://aquarius.tw.rpi.edu/html/DA/group2/).

- You are asked to Explore, Inspect the code/scripts in the Rstudio environment and get familiar with those scripts.

- As you working on those given scripts, and your goal is to understand the R functions that are available in those scripts by using the help() function in Rstudio and searching the web.

Scripts – work through these

See in folder group2/ Lab3

Go over the following scrips,

Lab3_ctree1.R

Lab3_ctree2.R

Lab3_ctree3.R

script fragments in R available on the web site:
(aquarius.tw.rpi.edu/html/DA)

you are allowed to work in small groups and discuss.

Dataset Check-In

- Mandatory One-on-One with the instructor today to check-in your dataset (Dataset Documentation), if you have not met with instructor last Monday, you <u>must</u> meet with the instructor during the class time.
- **Reminder**: Assignment 5 is due Monday, (02/24/2020) and individual student presentations will take place during the class time.