

FIT3152 Data analytics. Tutorial 07:

Decision Trees

Objectives:

- Understand entropy and information gain in relation to decision tree algorithms
- Create decision trees using R
- Using decision trees for profiling
- Using decision trees for classifying unseen instances
- Evaluation of decision tree models – confusion matrices

Reference: An Introduction to Statistical Learning with applications in R (Springer Texts in Statistics), James, Witten, Hastie and Tibshirani, Chapter 8 (available on-line from Monash Library)

Pre-tutorial Activity

The “diamonds” data set comes packaged with ggplot2 and contains data about the price of diamonds as well as information on size and the 4 Cs affecting diamond price: carat (size), cut, colour and clarity.

1. Create a class variable named “price_level” as a factor and with two values “High” and ”Low” based on the median price of diamonds.

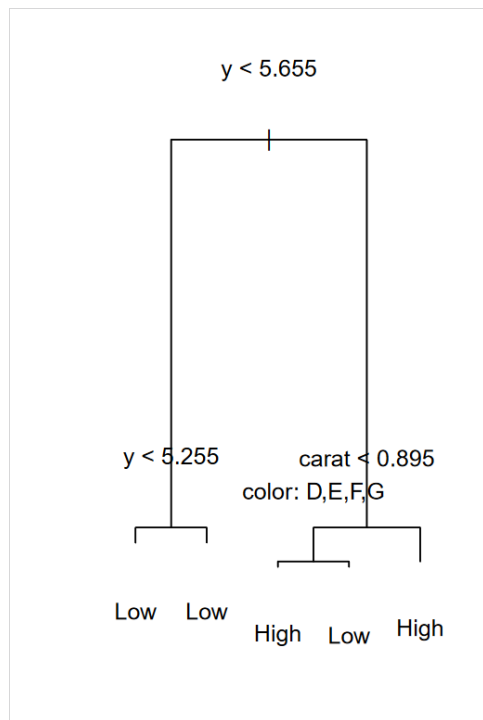
```
D = diamonds
price_level = ifelse(D$price >= median(D$price), "High", "Low")
D = cbind(D, as.factor(price_level))
colnames(D)[11] = "price_level"
```

2. Split the data into 70% training and 30% testing data sets. Fit a decision tree model to predict the price_level of diamonds and check accuracy using a confusion matrix.

```
# partition on to 70% training and 30% test data
set.seed(9999)
train.row = sample(1:nrow(D), 0.7*nrow(D))
D.train = D[train.row,]
D.test = D[-train.row,]

# fit decision tree model to predict price_level
library(tree)
D.fit=tree(price_level ~ .-price, data=D.train)

summary(D.fit)
plot(D.fit)
text(D.fit, pretty = 0)
```



Output:

Classification tree:

```
tree(formula = price_level ~ . - price, data = D.train)
```

Variables actually used in tree construction:

```
[1] "y"      "carat" "color"
```

Number of terminal nodes: 5

Residual mean deviance: 0.2438 = 9205 / 37750

Misclassification error rate: 0.04733 = 1787 / 37758

```
# create confusion matrix using prediction and check accuracy
```

```
D.predict = predict(D.fit, D.test, type = "class")
```

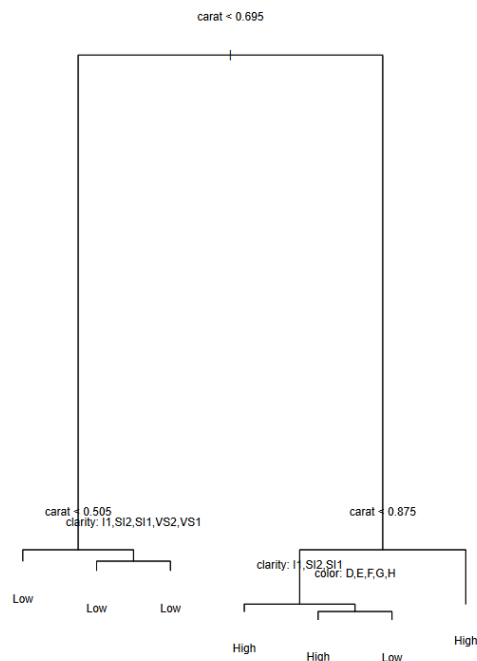
```
table(actual = D.test$price_level, predicted = D.predict)
```

	predicted	
actual	High	Low
High	7635	497
Low	283	7767

- Next, fit a decision tree model to predict price_level using the 4 C variables (carat (size), cut, colour and clarity) only and compare accuracy with the previous model.

```
# fit tree model using the 4C variables only
library(tree)
DC.fit=tree(price_level ~ carat + cut + color + clarity, data=D.train)

summary(DC.fit)
plot(DC.fit)
text(DC.fit, pretty = 0)
```



```
Classification tree:
tree(formula = price_level ~ carat + cut + color + clarity, data = D.train)
Variables actually used in tree construction:
[1] "carat" "clarity" "color"
Number of terminal nodes: 7
Residual mean deviance: 0.2063 = 7787 / 37750
Misclassification error rate: 0.05021 = 1896 / 37758
```

```
# create confusion matrix using prediction and check accuracy
DC.predict = predict(DC.fit, D.test, type = "class")
table(actual = D.test$price_level, predicted = DC.predict)
```

```
      predicted
actual High  Low
High  7871  261
Low   592  7458
```

Tutorial Activities

- Work through the examples in the lecture slides. For the examples using R you will need to download and install the ‘tree’ package using the following code.

```
install.packages("tree")
library(tree)
```

5 Calculation of entropy and information gain

Table 1 below includes data for 10 different types of aliens. The data is to be used to determine which aliens are friendly and which are not.

- What is the entropy of the IsFriendly class?
- Which decision attribute should you choose as the root of the decision tree – you can work this out without doing any calculations. Explain why you chose that attribute.
- What is the information gain of the attribute you chose in b.?
- Using the attribute you chose in b. as the root node, and using examples 1 to 10, build a decision tree for classifying aliens as friendly or not.
- Using your decision tree, what are the predictions for aliens 11, 12, 13 in table 2?

Table 1: Training Set

ID	colour	size	teeth	IsFriendly
1	red	medium	yes	no
2	blue	big	no	yes
3	green	medium	no	no
4	green	small	yes	no
5	blue	big	yes	yes
6	blue	small	yes	yes
7	red	small	no	yes
8	red	medium	no	yes
9	blue	medium	yes	yes
10	green	small	no	no

Table 2: Test Set

ID	colour	size	teeth	IsFriendly
11	red	big	yes	?
12	green	big	yes	?
13	blue	small	no	?

Calculations for decision tree using lecture template

For Aliens Data			Log Base	2	0.3010		
Initial State	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Entropy(S)	6	4	0.6000	-0.7370	0.4000	-1.3219	0.9710
colour	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
red	2	1	0.6667	-0.5850	0.3333	-1.5850	0.9183
blue	4	0	1.0000	0.0000	0.0000	0.0000	0.0000
green	0	3	0.0000	0.0000	1.0000	0.0000	0.0000
EEntropy(colour)							0.2755
Gain(S, colour)							0.6955
size	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
medium	2	2	0.5000	-1.0000	0.5000	-1.0000	1.0000
big	2	0	1.0000	0.0000	0.0000	0.0000	0.0000
small	2	2	0.5000	-1.0000	0.5000	-1.0000	1.0000
EEntropy(size)							0.8000
Gain(S, size)							0.1710
teeth	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
yes	3	2	0.6000	-0.7370	0.4000	-1.3219	0.9710
no	2	3	0.4000	-1.3219	0.6000	-0.7370	0.9710
Eentropy(teeth)							0.9710
Gain(S, teeth)							0.0000
After splitting on colour: blue and green branches terminate as leaf nodes. Evaluate red branch:							
red, size	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
medium	1	1	0.5000	-1.0000	0.5000	-1.0000	1.0000
small	1	0	1.0000	0.0000	0.0000	0.0000	0.0000
EEntropy(size)							0.6667
Gain(S, size)							0.2516
red, teeth	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
yes	0	1	0.0000	0.0000	1.0000	0.0000	0.0000
no	2	0	1.0000	0.0000	0.0000	0.0000	0.0000
Eentropy(teeth)							0.0000
Gain(S, teeth)							0.9183
So split on teeth and tree finished.							

- 6 The built-in data set `mtcars` describes the fuel consumption and 10 other variables for 32 cars produced during 1973 – 1974. Fuel consumption is determined as miles per gallon (`mpg`). Create a decision tree to classify cars as either high consumption (greater than the median), or low consumption. To do this, follow the steps below.

- a. Convert the `mpg` variables in to a class using the script below and create new data set: `carsclass`.

```
data(mtcars)
attach(mtcars)
summary(mpg)
cons = ifelse(mpg >= 19.20, "yes", "no")
cons = as.factor(cons)
carsclass = cbind(cons, mtcars)
head(carsclass)
```

- b. Partition your new data set into 70% training and 30% test.

```
# partition on to training and test data
set.seed(9999)
train.row = sample(1:nrow(carsclass), 0.7*nrow(carsclass))
c.train = carsclass[train.row,]
c.test = carsclass[-train.row,]
```

- c. Fit the ‘tree’ model to your data. Make sure you don’t include `mpg` as an attribute. You may need to first create a synthetically larger training data set using resampling with replacement as was done for the `playtennis` example.

```
# get larger training data set
set.seed(9999)
cl.train = c.train[sample(nrow(c.train), 100, replace = TRUE),]

# fit tree model
library(tree)
c.fit=tree(cons~.-mpg, data=cl.train)
```

- d. Examine your decision tree using `summary` and `plot` functions. What are the important attributes for determining fuel economy?

```
summary(c.fit)
plot(c.fit)
text(c.fit, pretty = 0)
```

- e. Using the test data set, calculate the accuracy of your model. How well does it predict fuel economy?

```
# test accuracy
c.predict = predict(c.fit, c.test, type = "class")
table(actual = c.test$cons, predicted = c.predict)
```

Outputs

```
> summary(c.fit)
```

```
Classification tree:
```

```
tree(formula = cons ~ . - mpg, data = cl.train)
```

Variables actually used in tree construction:

```
[1] "disp" "cyl" "qsec"
```

Number of terminal nodes: 4

Residual mean deviance: 0.06773 = 6.502 / 96

Misclassification error rate: 0.01 = 1 / 100

This shows that displacement, number of cylinders and "qsec", time over a quarter of a mile are the important attributes for building the tree. Misclassification rate on the training data is 2/100 or 2%

```
> table(actual = c.test$cons, predicted = c.predict)
      predicted
actual no yes
   no    7   0
   yes    1   2
> (7 + 2)/10
[1] 0.9
```

Accuracy on the test data is 90%

- 7 The Zoo data set (zoo.data.csv) contains data relating to seven classes of animals.

Using all the data, construct a decision tree to predict class "type" based on the other attributes. Note that you will need to specify that "type" is a factor. Which attributes are most influential in determining the class of an animal. What classes have less than 100% accuracy?

```
# set working directory to desktop
setwd("~/Desktop")
# clean up the environment before starting
rm(list = ls())

# read in data
zoo=read.csv("zoo.data.csv", stringsAsFactors = TRUE)
zoo[,2:18]=lapply(zoo[,2:18] , factor)

# fit tree model
library(tree)
z.fit=tree(type~.-animal_name, data = zoo)
summary(z.fit)
plot(z.fit)
text(z.fit, pretty=0)
```

Outputs

```
> summary(z.fit)
```

Classification tree:

```
tree(formula = type ~ . - animal_name, data = zoo)
```

Variables actually used in tree construction:

```
[1] "milk"      "feathers" "backbone" "legs"      "fins"
```

Number of terminal nodes: 6

Residual mean deviance: 0.2355 = 22.37 / 95

Misclassification error rate: 0.05941 = 6 / 10

The accuracy on whole data set (training) is 1-(6/101)=0.94 or 94%
You can see the important attributes in the summary.

Now split your data into a training set 70% and test set 30% and refit the model. What is the overall accuracy of the model when measured on the test set?

```
zt.fit = tree(formula = type ~ . -animal_name, data = z.train)
summary(zt.fit)
```

```
z.predict = predict(zt.fit, z.test, type="class")
table(actual = z.test$type, predicted = z.predict)
```

Outputs

```
summary(zt.fit)
```

```
Classification tree:
tree(formula = type ~ . - animal_name, data = z.train)
Variables actually used in tree construction:
[1] "milk"      "backbone" "legs"      "feathers"  "fins"
Number of terminal nodes: 6
Residual mean deviance: 0.2135 = 13.67 / 64
Misclassification error rate: 0.04286 = 3 / 70
```

```
# The accuracy on test data is 1-(3/70)=0.95 or 95%
```

```
Important attributes are mile, tail, aquatic, feathers
```

```
> table(actual = z.test$type, predicted = z.predict)
```

	predicted						
actual	1	2	3	4	5	6	7
1	12	0	0	0	0	0	0
2	0	8	0	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	0	5	0	0	0
5	0	0	2	0	0	0	0
6	0	0	0	0	0	1	0
7	0	0	0	0	0	1	1

```
> (12+8+1+5+1+1)/31
[1] 0.9032258
```

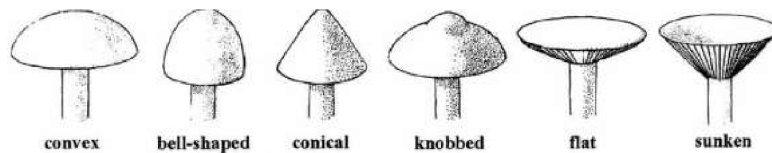
- 8 The Mushroom data set (mushroom.data.csv) contains 22 pieces of information about 8000+ species of mushrooms. This data set was obtained from the UCI Machine Learning Repository. Further information about the data can be obtained from: <https://archive.ics.uci.edu/ml/datasets/Mushroom>

Using the data, construct a decision tree to predict whether an unclassified mushroom is of “class” poisonous or edible based on the other attributes:

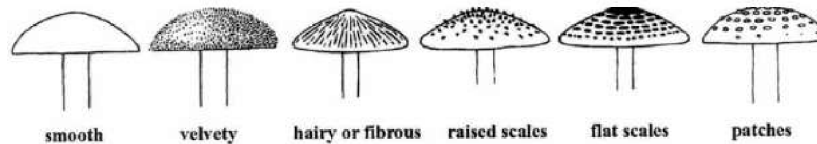
1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
- ...

You should create a training and test data set and report the accuracy of your model. Which attributes are most important in distinguishing between poisonous and edible mushrooms?

Mushroom cap shapes



Mushroom cap surfaces



```
# read in data
m = read.csv("mushroom.data.csv", stringsAsFactors = TRUE)

# partition on to training and test data
set.seed(9999)
train.row = sample(1:nrow(m), 0.7*nrow(m))
m.train = m[train.row,]
m.test = m[-train.row,]

# fit tree model on training data
library(tree)
m.fit=tree(class~., data = m.train)
summary(m.fit)
plot(m.fit)
text(m.fit, pretty=0)

# test accuracy
m.predict = predict(m.fit, m.test, type = "class")
table(actual = m.test$class, predicted = m.predict)
```

Outputs

```
> summary(m.fit)

Classification tree:
tree(formula = class ~ ., data = m.train)
Variables actually used in tree construction:
[1]
"odor"                                "spore.print.color"          "stalk.color.below.r
ing"
Number of terminal nodes:  4
Residual mean deviance:  0.02936 = 166.8 / 5682
Misclassification error rate: 0.002286 = 13 / 5686

> plot(m.fit)
> text(m.fit, pretty=0)

> #test accuracy
> m.predict = predict(m.fit, m.test, type = "class")

> table(actual = m.test$class, predicted = m.predict)
      predicted
actual    e    p
   e 1287    0
   p   11 1140
```

- 9 Using data from the Kaggle competition: Titanic: Machine Learning from Disaster (ref <https://www.kaggle.com/c/titanic/data>) develop a decision tree model to predict whether a passenger would have or have not survived. The data has been portioned into a training and a test set: (Kaggle.Titanic.train.csv, Kaggle.Titanic.test.csv) The main details of the attributes, from the Kaggle site, are:

VARIABLE DESCRIPTIONS:

survival	Survival (0 = No; 1 = Yes)
pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
name	Name
sex	Sex
age	Age
sibsp	Number of Siblings/Spouses Aboard
parch	Number of Parents/Children Aboard
ticket	Ticket Number
fare	Passenger Fare
cabin	Cabin
embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

How accurate is your model? Based on your model, what are the most important predictors for survival?

```
# clean up the environment before starting
rm(list = ls())

#read in data
t.train=read.csv("Kaggle.Titanic.train.csv")
t.test=read.csv("Kaggle.Titanic.test.csv")

t.train$PassengerId=NULL
t.train$Ticket=NULL
t.train$Name=NULL
t.test$PassengerId=NULL
t.test$Ticket=NULL
t.test$Name=NULL

#Make Survived a factor (otherwise model fits a regression tree)
t.train$Survived = as.factor(t.train$Survived)

#fit tree model on training data
library(tree)
t.fit=tree(Survived~.-Cabin, data=t.train)
summary(t.fit)
plot(t.fit)
text(t.fit, pretty=0)

#tpredict = predict(t.fit, t.test, type = "vector")
#tpredict = predict(t.fit, t.test, type = "class")

> summary(t.fit)

Classification tree:
tree(formula = Survived ~ . - Cabin, data = t.train)
Variables actually used in tree construction:
[1] "Sex" "Pclass" "Fare" "Age" "SibSp"
Number of terminal nodes: 7
Residual mean deviance: 0.7975 = 563.9 / 707
Misclassification error rate: 0.1737 = 124 / 714
```