DSA1101 Topic 5: Decision Trees

Dawn Cheung

2024-03-04

Bank Sample Dataset

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.4     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ 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 be
come errors
```

```
library(class)
```

```
## Warning: package 'class' was built under R version 4.3.2
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.3.2
```

```
library(rpart.plot) #remember its a diff package as rpart!!
```

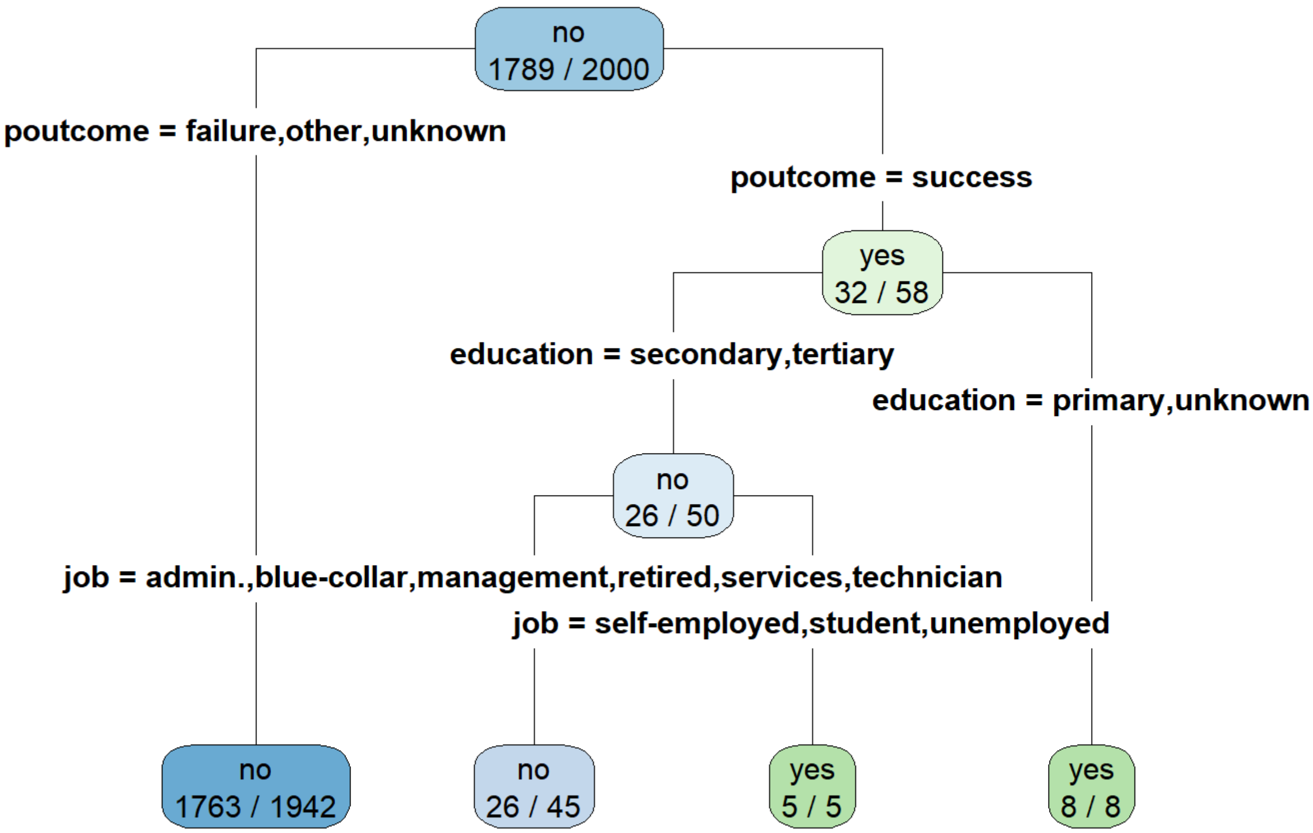
```
## Warning: package 'rpart.plot' was built under R version 4.3.2
```

```
bankdata = read.csv("~/Github/DSA1101 Slayers/datasets/bank-sample.csv")
attach(bankdata)

#response variable: subscribed (yes/no)
#we have 16 features => not using all (we will use 8)

fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + po
utcome,
             method = "class", #tells R to return the category of response (yes or no)
             data = bankdata,
             control = rpart.control(minsplit = 1), #tells R how to split the node; minsplit
= 1 means a branch is created when there is at least 1 observation in that branch => tells R
how big we want the tree to be
             parms = list(split = "information")) # what criteria to use to choose the root n
ode & internal nodes; either info gained or gini index; default val is gini

summary(fit)
```

```
## Call:
## rpart(formula = subscribed ~ job + marital + education + default +
##     housing + loan + contact + poutcome, data = bankdata, method = "class",
##     parms = list(split = "information"), control = rpart.control(minsplit = 1))
##   n= 2000
##
##           CP nsplit rel error    xerror       xstd
## 1 0.02843602      0 1.0000000 1.0000000 0.06511019
## 2 0.01658768      1 0.9715640 1.0331754 0.06605180
## 3 0.01000000      3 0.9383886 0.9810427 0.06456213
##
## Variable importance
##  poutcome education       job
##        80        11         9
##
## Node number 1: 2000 observations,    complexity param=0.02843602
##   predicted class=no   expected loss=0.1055  P(node) =1
##     class counts:  1789   211
##    probabilities: 0.895 0.105
##   left son=2 (1942 obs) right son=3 (58 obs)
##   Primary splits:
##       poutcome  splits as  LLRL,         improve=36.876590, (0 missing)
##       contact   splits as  RRL,          improve=25.971290, (0 missing)
##       housing   splits as  RL,           improve=18.328410, (0 missing)
##       job       splits as  LLLLLRLLRLRR, improve= 9.052902, (0 missing)
##       education splits as  LLRR,         improve= 3.328298, (0 missing)
##
## Node number 2: 1942 observations
##   predicted class=no   expected loss=0.09217302  P(node) =0.971
##     class counts:  1763   179
##    probabilities: 0.908 0.092
##
## Node number 3: 58 observations,    complexity param=0.01658768
##   predicted class=yes  expected loss=0.4482759  P(node) =0.029
##     class counts:    26    32
##    probabilities: 0.448 0.552
##   left son=6 (50 obs) right son=7 (8 obs)
##   Primary splits:
##       education splits as  RLLR,         improve=5.2742870, (0 missing)
##       job       splits as  LL--LLRLRLR-, improve=3.8479820, (0 missing)
##       housing   splits as  RL,           improve=1.9292220, (0 missing)
##       contact   splits as  RRL,          improve=0.8131580, (0 missing)
##       loan      splits as  LR,           improve=0.6018268, (0 missing)
##
## Node number 6: 50 observations,    complexity param=0.01658768
##   predicted class=no   expected loss=0.48  P(node) =0.025
##     class counts:    26    24
##    probabilities: 0.520 0.480
##   left son=12 (45 obs) right son=13 (5 obs)
##   Primary splits:
##       job     splits as  LL--LLRLRLR-, improve=3.9723870, (0 missing)
##       contact splits as  RLL,          improve=2.7760700, (0 missing)
##       housing splits as  RL,           improve=1.3488020, (0 missing)
##       loan    splits as  LR,           improve=0.7450307, (0 missing)
##       marital splits as  LLR,          improve=0.3260181, (0 missing)
```

```
##
## Node number 7: 8 observations
##   predicted class=yes  expected loss=0  P(node) =0.004
##     class counts:     0     8
##    probabilities: 0.000 1.000
##
## Node number 12: 45 observations
##   predicted class=no   expected loss=0.4222222  P(node) =0.0225
##     class counts:    26    19
##    probabilities: 0.578 0.422
##
## Node number 13: 5 observations
##   predicted class=yes  expected loss=0  P(node) =0.0025
##     class counts:     0     5
##    probabilities: 0.000 1.000
```

```
#to fit the plotted tree:
rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE)
```



Ok notice the first node shown is a modal category of the response ie it starts with a no (which is the majority)

So not the root node (which needs to be an input vairable)

Choosing the root node

Why was poutcome selected as the decsion variable at the root node?

i.e. why was poutcome chosen by the algorithm to be the first split? finding the most useful feature in the dataset to add to the tree

Selecting the most informative attribute based on 2 basic measures:

- Entropy: the impurity of an attribute
- Information gain: the reduction in purity should a split be made there

Purity

Its probability of the corresponding class

- When only considering the particular response variable, ignoring all other attributes/features, then the probability of [the response variable being equal to something] is the purity
- eg it is 89.45% pure on the class where [the response variable is equal to something], and the rest, 10.55% pure on the class [variable equals to smth else]

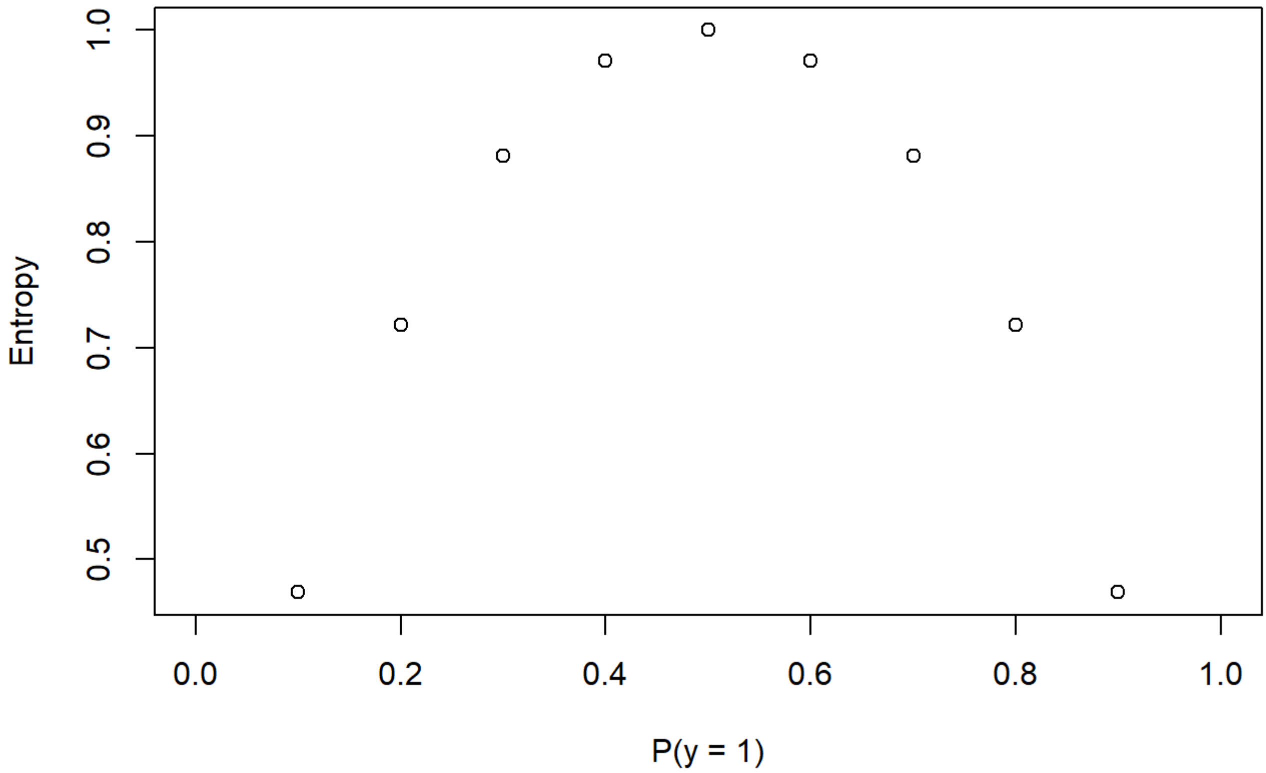
Entropy

- Given variable Y and and the set of possible categorical values it can take, $(y_1, y_2, \dots, y_K)$, the entropy of Y is defined as

$$D_Y = -\sum_{j=1}^{K} \underbrace{P(Y = y_j) \log_2 P(Y = y_j)}, $$

where $P(Y = y_j)$ denotes the purity or the probability of the class $Y = y_j$, and $\sum_{j=1}^{K} P(Y = y_j) = 1.$

```
p = seq(0, 1, 0.1)
Entropy = -(p*log2(p) + (1-p)*log2(1-p))
plot(p, Entropy, ylab = "Entropy", xlab = "P(y = 1)")
```

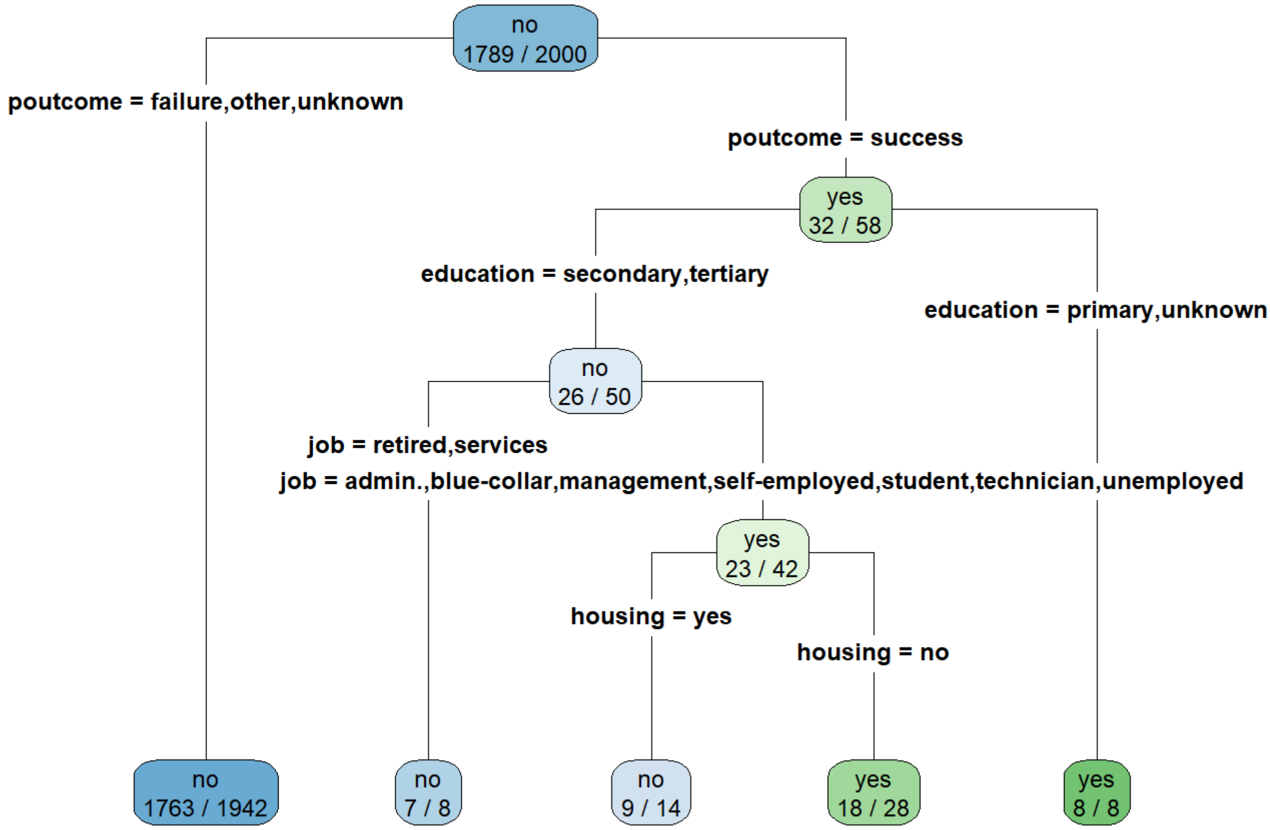


Traversing down the tree, how are the subsequent decision variables at each node selected?

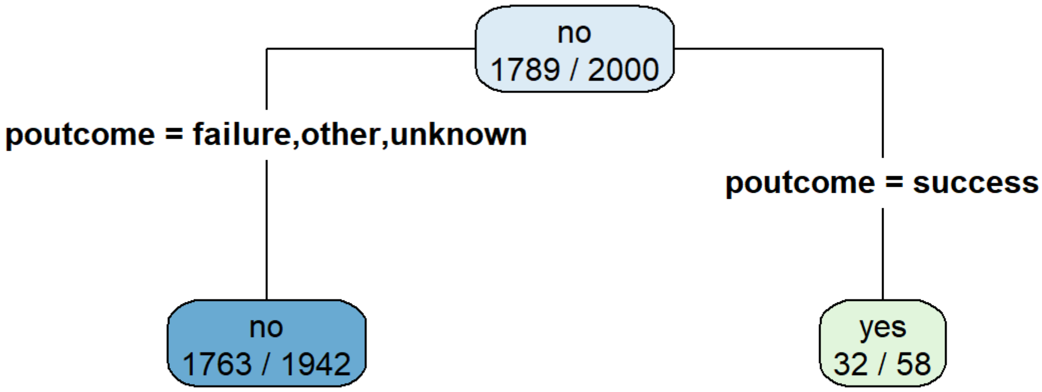
i.e. understanding how the algorithm decides what should be a branch and what should not be made into a branch

Playing with maxdepth, minsplit, cp (complexity parameter) in control = rpart.control()

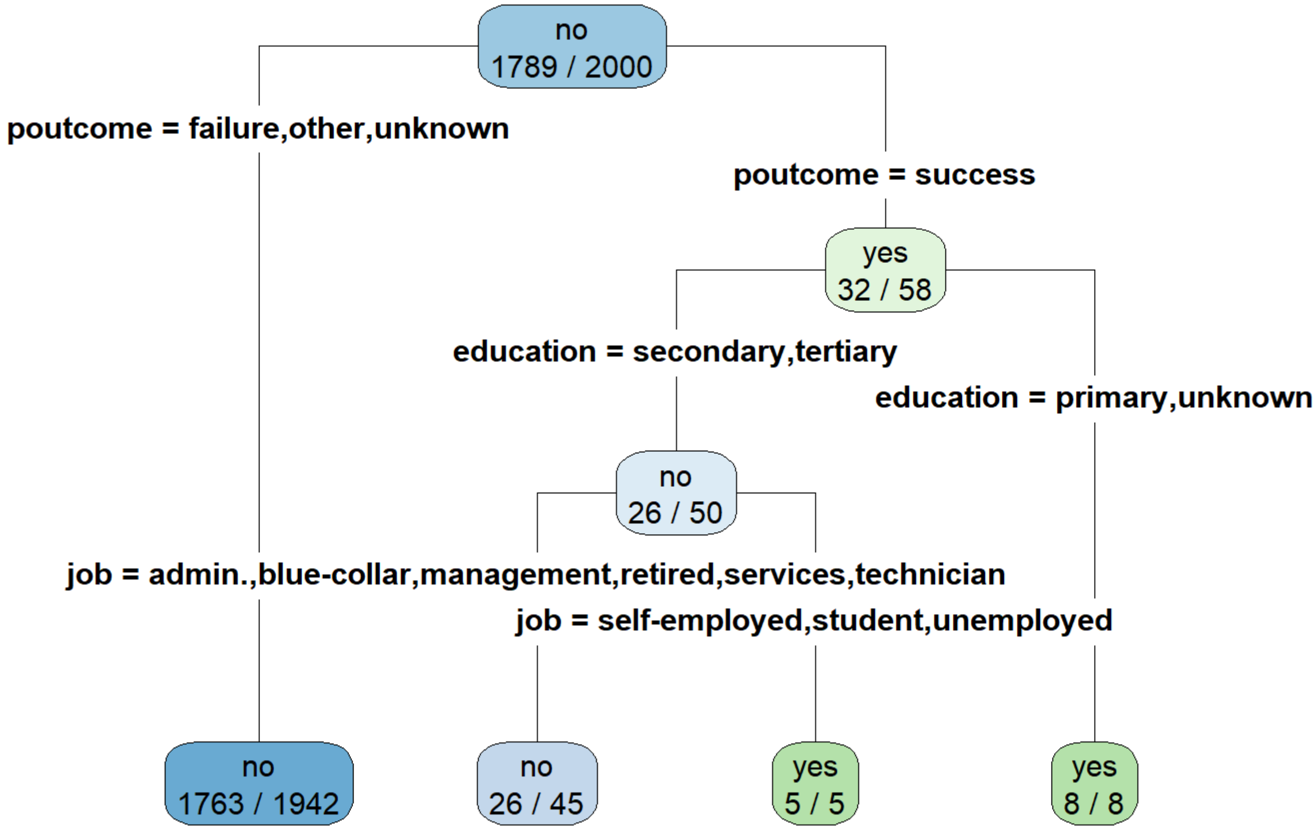
```
#maxdepth = 4:
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + po
utcome,
             method = "class", #tells R to return the category of response (yes or no)
             data = bankdata,
             control = rpart.control(maxdepth = 4), #tells R how to split the node; minsplit
= 1 means a branch is created when there is at least 1 observation in that branch => tells R
how big we want the tree to be
             parms = list(split = "information"))

rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE)
```



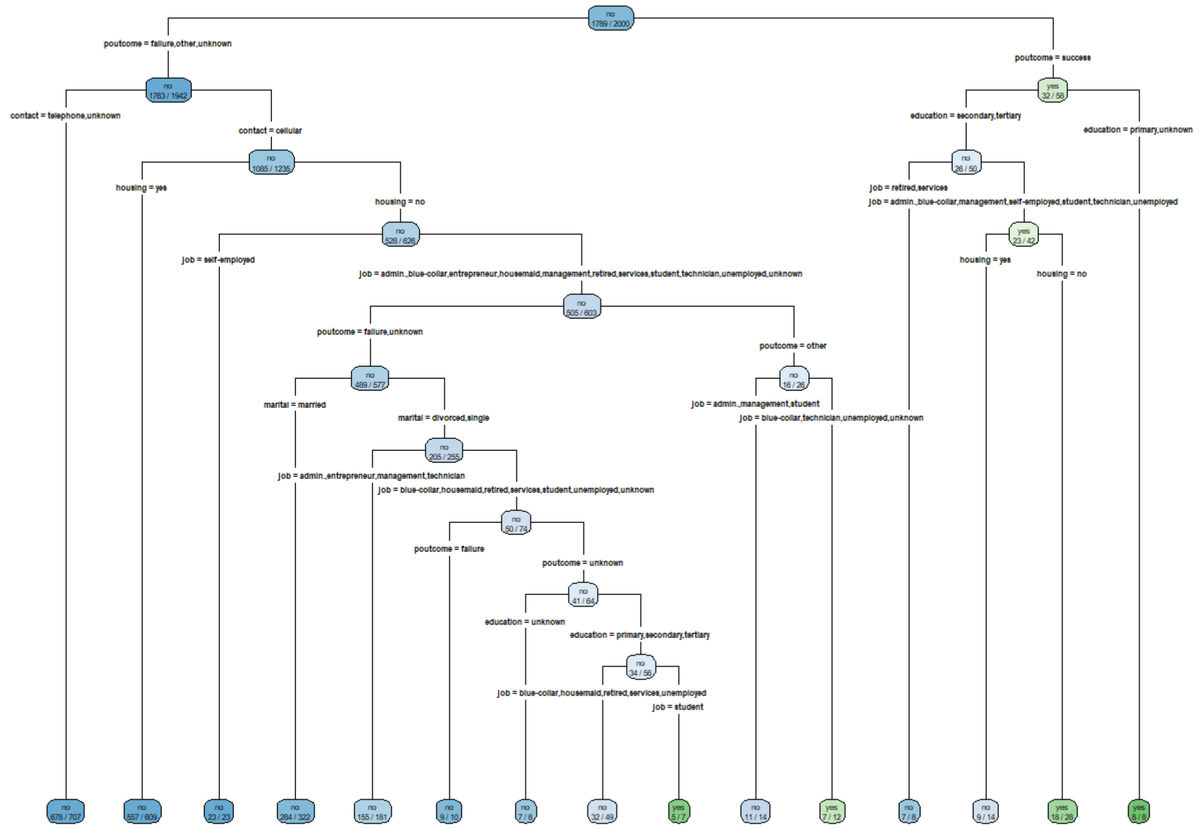

```
#maxdepth = 1:
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + po
utcome,
             method = "class", #tells R to return the category of response (yes or no)
             data = bankdata,
             control = rpart.control(maxdepth = 1), #tells R how to split the node; minsplit
= 1 means a branch is created when there is at least 1 observation in that branch => tells R
how big we want the tree to be
             parms = list(split = "information"))

rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE)
```




```
#minsplit = 15:
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + po
utcome,
             method = "class", #tells R to return the category of response (yes or no)
             data = bankdata,
             control = rpart.control(minsplit = 15), #tells R how to split the node; minsplit
= 1 means a branch is created when there is at least 1 observation in that branch => tells R
how big we want the tree to be
             parms = list(split = "information")) # what criteria to use to choose the root n
ode & internal nodes; either info gained or gini index; default val is gini

#to fit the plotted tree:
rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE)
```




```
#cp = 0.001
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + po
utcome,
             method = "class", #tells R to return the category of response (yes or no)
             data = bankdata,
             control = rpart.control(cp = 0.001), #tells R how to split the node; minsplit =
1 means a branch is created when there is at least 1 observation in that branch => tells R ho
w big we want the tree to be
             parms = list(split = "information")) # what criteria to use to choose the root n
ode & internal nodes; either info gained or gini index; default val is gini
rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE)
```



As maxdepth increases, tree becomes larger.

As minsplit increases, tree becomes smaller.

As cp decreases, tree becomes larger. [default is cp = 0.01]

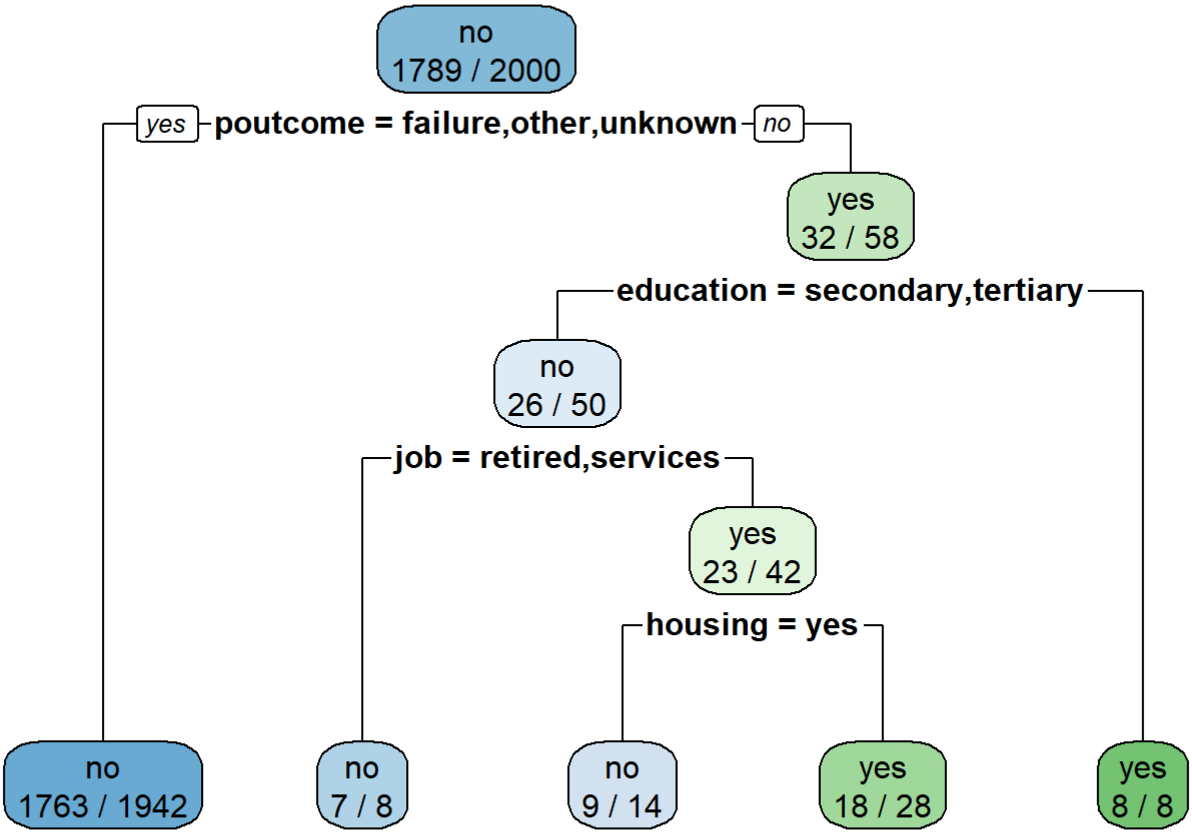
How to test which parameters are best? Just use for loops to try each parameter and attain accuracy lol

Playing with varlen, faclen, clip.right.labs, type in rpart.plot()

- varlen = variable length
  - 0: full name will be written
- faclen = factor length
  - 0: full name will be written
- clip.right.labs literally means the name of the variable wont be repeated on the right branch
  - TRUE or FALSE

MEANING OF THE TREE WILL REMAIN THE SAME

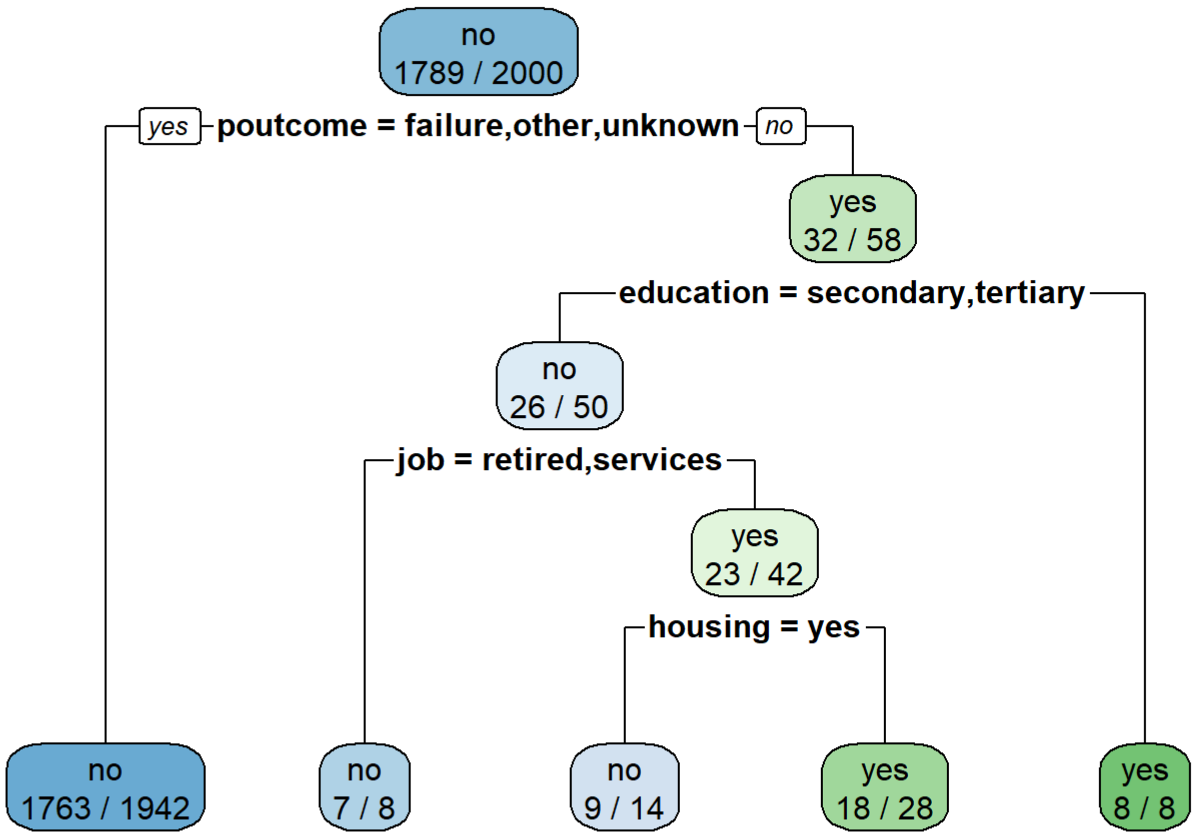
```
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + po
utcome,
             method = "class", #tells R to return the category of response (yes or no)
             data = bankdata,
             control = rpart.control(maxdepth = 4), #tells R how to split the node; minsplit
= 1 means a branch is created when there is at least 1 observation in that branch => tells R
how big we want the tree to be
             parms = list(split = "information"))

#type = 2:
rpart.plot(fit, type = 2, extra = 2, clip.right.labs = FALSE)
```



Analysis of Fitted Tree

modal catagory => the major/majority catagory

```
fit <- rpart(subscribed ~ job + marital + education + default + housing + loan + contact + po
utcome,
             method = "class", #tells R to return the category of response (yes or no)
             data = bankdata,
             control = rpart.control(maxdepth = 4), #tells R how to split the node; minsplit
= 1 means a branch is created when there is at least 1 observation in that branch => tells R
how big we want the tree to be
             parms = list(split = "information"))

#type = 2:
rpart.plot(fit, type = 2, extra = 2, clip.right.labs = FALSE)
```



how model decides how to structure the tree: by splitting at this depth, it reduces the entropy the most

DTdata.csv [play_decsion]

```
library(rpart)
library(rpart.plot) #remember its a diff package as rpart!!
play_decsion = read.csv("~/Github/DSA1101 Slayers/datasets/DTdata.csv")
attach(play_decsion)

newdata = data.frame(Outlook = "rainy", Temperature = "mild", Humidity = "")
```

how model decides how to structure the tree: by splitting at this depth, it reduces the entropy the most

TUTORIAL QUESTIONS

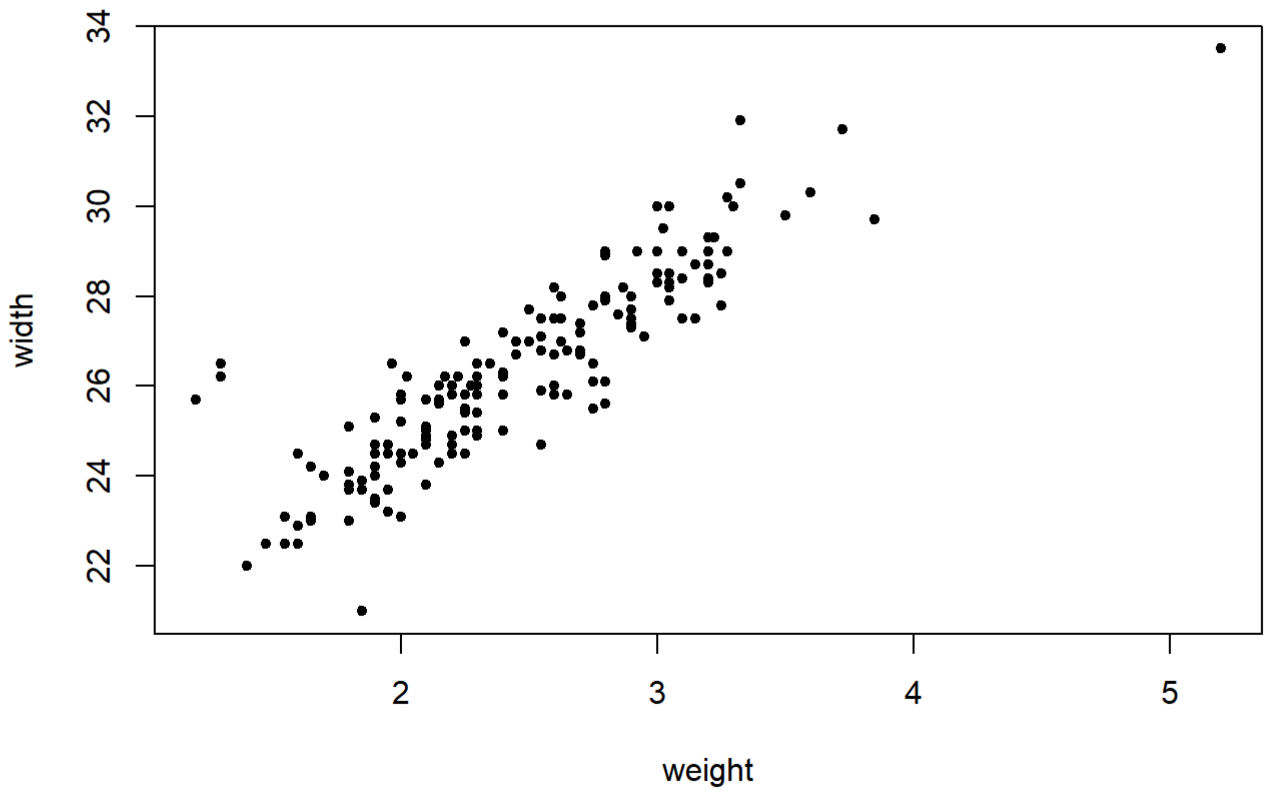
(MLR) Consider the horseshoe female crab data given in the csv file crab.csv. We would want to form a model for the weight of the female crabs (kg), which depends on its width (cm) and its spine condition (1 = both good, 2 = one worn or broken, 3 = both worn or broken).

a. Produce a scatter plot of variable weight against width for different condition of spine.

```
library(tidyverse)

crabdata = read.csv("~/Github/DSA1101 Slayers/datasets/crab.csv")

#REMEMBER that when converting to factor/catagorical, address it as part of he crabdata datas
et, so dataset$variable
crabdata$spine  = as.factor(crabdata$spine)
attach(crabdata)
glimpse(crabdata)
```

```
## Rows: 173
## Columns: 5
## $ color  <int> 3, 4, 2, 4, 4, 3, 2, 4, 3, 4, 4, 3, 3, 5, 3, 2, 3, 3, 5, 3, 3, …
## $ spine  <fct> 3, 3, 1, 3, 3, 3, 1, 2, 1, 3, 3, 3, 3, 2, 1, 1, 3, 3, 3, 3, 2, …
## $ width  <dbl> 28.3, 22.5, 26.0, 24.8, 26.0, 23.8, 26.5, 24.7, 23.7, 25.6, 24.…
## $ satell <int> 8, 0, 9, 0, 4, 0, 0, 0, 0, 0, 0, 0, 11, 0, 14, 8, 1, 1, 0, 5, 4…
## $ weight <dbl> 3.05, 1.55, 2.30, 2.10, 2.60, 2.10, 2.35, 1.90, 1.95, 2.15, 2.1…
```

```
plot(weight, width, pch = 20)
```



b. Fit a linear regression model for weight which has two explanatories, width and spine.

```
M1 = lm(weight ~ width + spine, crabdata)
summary(M1)
```

```
##
## Call:
## lm(formula = weight ~ width + spine, data = crabdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.23016 -0.10828  0.01016  0.13356  0.96350
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.92955    0.27506 -14.286   <2e-16 ***
## width        0.24376    0.01002  24.335   <2e-16 ***
## spine2       0.05544    0.08475   0.654    0.514
## spine3      -0.06969    0.05065  -1.376    0.171
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2656 on 169 degrees of freedom
## Multiple R-squared:  0.7918, Adjusted R-squared:  0.7881
## F-statistic: 214.2 on 3 and 169 DF,  p-value: < 2.2e-16
```

c. Is the fitted model significant? Yes

When looking at the F-statistic's p-value, it is extremely small, $p < 2.2e-16$. Hence, compared to a model with no regressors (such that it is a straight horizontal line), the model in (b) does significantly better

d. Derive R2 and adjusted R2 of the fitted model.

```
paste0("R2 is ", summary(M1)$r.squared)
```

```
## [1] "R2 is 0.791759758590367"
```

```
paste0("Adjusted R2 is ", summary(M1)$adj.r.squared)
```

```
## [1] "Adjusted R2 is 0.788063186257652"
```

e. Write down the fitted model

```
paste0("y hat  = ", M1$coeff[0], "+ ", M1$coeff[1], "width + ", M1$coeff[2],"I(spine  = 2) +
", M1$coeff[3], "I(spine  = 3)")
```

```
## [1] "y hat  = + -3.92955229473623width + 0.243762770317257I(spine  = 2) + 0.05544486069152
22I(spine  = 3)"
```

f. Two female crabs of the same width, find the difference of their weight if one has spines are of good condition and another one with broken spines

```
0.05544 #?? not sure
```

```
## [1] 0.05544
```

g. Predict the weight of a female crab that has width of 27 cm and has both spines worn or broken.

```
q = data.frame(width = c(27), spine = c("3"))

predict(M1, newdata = q)
```

```
##        1
## 2.582352
```

The K-nearest neighbor classfier The table below provides a training data set containing six observations, three predictors, and one qualitative response variable, Y .

```
Obs X1 X2 X3 Y
1    0  3  0  Red
2    2  0  0  Red
3    0  1  3  Red
4    0  1  2  Green
5   -1  0  1  Green
6    1  1  1  Red
```

Suppose we wish to use this data set to make a prediction for Y when X1 = X2 = X3 = 0 using K-nearest neighbors.

a. Compute the Euclidean distance between each observation and the test point, X1 = X2 = X3 = 0.

```
Q = cbind(X1 = c(0, 2, 0, 0 ,-1, 1),
      X2 = c(3, 0, 1, 1 , 0, 1),
      X3 = c(0, 0, 3, 2 , 1, 1))

for (i in 1:6) {
  Ed = sqrt( (Q[[i, 1]]-0)^2 + (Q[[i, 2]]-0)^2 + (Q[[i, 3]]-0)^2 )
  print(Ed)
}
```

```
## [1] 3
## [1] 2
## [1] 3.162278
## [1] 2.236068
## [1] 1.414214
## [1] 1.732051
```

b. What is our prediction with K = 1? Why?

```r
library(class)
library(dplyr)

#train.x is Q

test.x = cbind(X1 = c(0),
      X2 = c(0),
      X3 = c(0))

train.y = c("Red", "Red", "Red", "Green", "Green", "Red")

knn.pred = knn(Q, test.x, train.y, k = 1)

knn.pred
```

```
## [1] Green
## Levels: Green Red
```

```r
#predict not needed here
```

c. What is our prediction with K = 3? Why?

```r
knn.pred = knn(Q, test.x, train.y, k = 3)

knn.pred
```

```
## [1] Red
## Levels: Green Red
```

```r
#WHAT DO YOU MEAN WHY
```

d. If the Bayes decision boundary (the gold standard decision boundary) in this problem is highly non-linear, then would we expect the best value for K to be large or small? Why?

Small k. Non linear means boundary line would be very flexible since it is influenced by local features of a handful of traning data points.

Measures of classifier performance

Suppose we have developed a K-nearest neighbors classifier for predicting diabetes status. The following table shows the actual response Y (1 =yes, 0 =no) and fitted value Yb using the classifier for 10 test data points. A test data point is predicted to be Gb = 1 if Y > δ ˆ , for a specified threshold value δ. (Recall that we use δ = 0.5 in class, also known as the majority rule).

We define

TPR = TP / TP + FN ; FPR = FP / FP + TN.

For each of the thresholds δ = 0.3, 0.6 and 0.8, *derive TPR and FPR* in making predictions with the K-nearest neighbors classifier for the 10 test data points. *Plot TPR against FPR for the three thresholds.*

```
Yi = c(1L, 1L, 1L, 1L, 0L, 0L, 1L, 0L, 0L)
Yi_hat = c(0.9, 0.5, 0.7, 0.4, 0.5, 0.2, 0.7, 0.9, 0.1, 0.1)




gginsane <- function(d) {
  TP = 0
  FP = 0
  TN = 0
  FN = 0
  for (i in 1:10) {
    if (Yi_hat[i] > d) {
      if (Yi[i] == 1) {
        TP = TP + 1
      }
      if (Yi[i] == 0L) {
        FN = FN + 1
      }
    } else if (Yi_hat[i] < d) {
      if (Yi[i] == 0L) {
        TN = TN + 1
        }
      if (Yi[i] == 1) {
        FP = FP + 1
        }
    }
  }
  TPR = TP/(TP + FN)
  FPR = FP/(FP + TN)
  print(TPR)
  print(FPR)
}

gginsane(0.3)
gginsane(0.6)
gginsane(0.8)
```

Error in if (Yi[i] == 0L) { : missing value where TRUE/FALSE needed for FUCKS SAKE

```r
Yi = c(1, 1, 0, 1, 1, 0, 0, 1, 0, 0)
Yi_hat = c(0.9, 0.5, 0.7, 0.4, 0.5, 0.2, 0.7, 0.9, 0.1, 0.1)



gginsane <- function(d) {
  for (i in 1:10) {
    if (Yi_hat > d && Yi == int(1)) {
      TP = TP + 1
    } else if (Yi_hat > d && Yi == int(0)) {
      FN = FN + 1
    } else if (Yi_hat < d && Yi == int(0)) {
      TN = TN + 1
    } else if (Yi_hat < d && Yi == int(1)) {
      FP = FP + 1
    }
  }
  TPR = TP / (TP + FN)
  FPR = FP / (FP + TN)
  return(TPR, FPR)
}

gginsane(0.3)
gginsane(0.6)
gginsane(0.8)
```

processing file: Topic5_MY-OWN-Rcode.Rmd

Quitting from lines 322-347 [3a] (Topic5_MY-OWN-Rcode.Rmd) Error in `Yi_hat > d && Yi == int(1)` : ! 'length = 10' in coercion to 'logical(1)' Backtrace: 1. global gginsane(0.3) Execution halted

i Yi Yˆi 1 1 0.9 2 1 0.5 3 0 0.7 4 1 0.4 5 1 0.5 6 0 0.2 7 0 0.7 8 1 0.9 9 0 0.1 10 0 0.1

b. Can we add the two points (0, 0) and (1, 1) to the plot of TPR against FPR in part (a). Explain why or why not

```r
#hello
```

4. The CSV file Caravan.csv contains data on 5822 real customer records on caravan insurance purchase. This data set is owned and supplied by the Dutch data mining company, Sentient Machine Research, and is based on real world business data. Each record consists of 86 variables, containing socio□demographic data (variables 1-43) and product ownership (variables 44-86). Variable 86 (Purchase) indicates whether the customer purchased a caravan insurance policy.

For this business, assume that the overall error rate (equivalently, the accuracy) is not of interest. Instead, the company wants to use the classi□er to predict who are the potential customers likely to purchase insurance. Then the metric precision will be important, since it relates the proportion of individuals who will actually purchase the insurance, among the group of individuals who are predicted to purchase insurance