CS 5900/STAT 46700 Topics in Data Science Spring 2025

Homework 3

[Vaishak Balachandra]

Q.N. 1) The result of 15 students enrolled in data mining course are provided in the table below. It also provides few other categorical variables:

- Course: whether enrolled in other courses (Yes/No)
- Background: whether student is from a Math, computer science (CS) or other background
- Working: whether student working (W) or not working (NW)

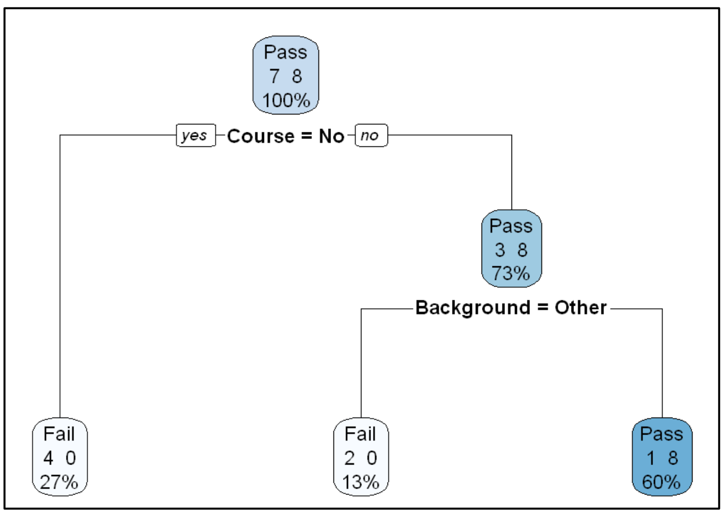
The target variable is Result a binary (Pass/Fail) variable and the other variables are predictor variables.

ID	Result	Course	Background	Working
1	Pass	Yes	Math	NW
2	Fail	No	Math	W
3	Fail	Yes	Math	W
4	Pass	Yes	CS	NW
5	Fail	No	Other	W
6	Fail	Yes	Other	W
7	Pass	Yes	Math	NW
8	Pass	Yes	CS	NW
9	Pass	Yes	Math	W
10	Pass	Yes	CS	W
11	Pass	Yes	CS	W
12	Pass	Yes	Math	NW
13	Fail	Yes	Other	W
14	Fail	No	Other	NW
15	Fail	No	Math	W

a) Calculate the entropy of the Result.
b) Identify the root node of the above data by calculating the information gain.
c) Construct a decision tree for the subject data using R.

```
> # q1
>
> q1 <- read.csv("q1.csv")
> head(q1)
  ID Result Course Background Working
1  1   Pass    Yes       Math      NW
2  2   Fail     No       Math       W
3  3   Fail    Yes       Math       W
4  4   Pass    Yes         CS      NW
5  5   Fail     No      Other       W
6  6   Fail    Yes      Other       W
> names(q1)
[1] "ID"         "Result"     "Course"     "Background" "Working"
```

```
> attach(q1)
>
> # a
> p = table(Result)
> install.packages("DescTools")
> library(DescTools)
> Entropy(p)
[1] 0.9967916
> cat("The entropy value of 0.9967 indicates a relatively high level of uncertainty in the Result v
ariable")
The entropy value of 0.9967 indicates a relatively high level of uncertainty in the Result variable
>
>
> # b
> Ent_Result = Entropy(table(Result))
> Ent_Result
[1] 0.9967916
> Ent_Result_Course <- sum(prop.table(table(q1$Course)) * sapply(unique(q1$Course), function(x) Ent
ropy(table(q1$Result[q1$Course == x]))))
> Ent_Result_Background <- sum(prop.table(table(q1$Background)) * sapply(unique(q1$Background), fun
ction(x) Entropy(table(q1$Result[q1$Background == x]))))
> Ent_Result_Working <- sum(prop.table(table(q1$Working)) * sapply(unique(q1$Working), function(x)
Entropy(table(q1$Result[q1$Working == x]))))
> IG_Result_Course = Ent_Result - Ent_Result_Course
> IG_Result_Background = Ent_Result - Ent_Result_Background
> IG_Result_Working = Ent_Result - Ent_Result_Working
> cat("Information gain:
+ 1. Course:" , IG_Result_Course,
+ "\n2. Background:", IG_Result_Background,
+ "\n3. Working:", IG_Result_Working)
Information gain:
1. Course: 0.7713647
2. Background: 0.7340641
3. Working: 0.1858052
> cat("Thus, here 'Course' is the root node!!")
Thus, here 'Course' is the root node!!
>
>
> # c
> install.packages("rpart")
> library("rpart")
> install.packages("rpart.plot")
> library("rpart.plot")
> model <- rpart(Result ~ Course + Background + Working, data = q1, method = "class",
+               control = rpart.control(cp = 0, minsplit = 2, minbucket = 1))
> rpart.plot(model, type = 2, extra = 101, box.palette = "Blues")
```



Q.N. 2) Consider the dataset `OJ` (Orange Juice) available in ISLR package. It describes the purchasing habit of the customer either purchased Citrus Hill or Minute Maid Orange Juice. Several characteristics of the customer and product are recorded.

a) Import the dataset in R and print the variable names.

b) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

c) Fit a tree to the training data, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. How many terminal nodes does the tree have?

d) Create a plot of the tree, and interpret the results.

e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

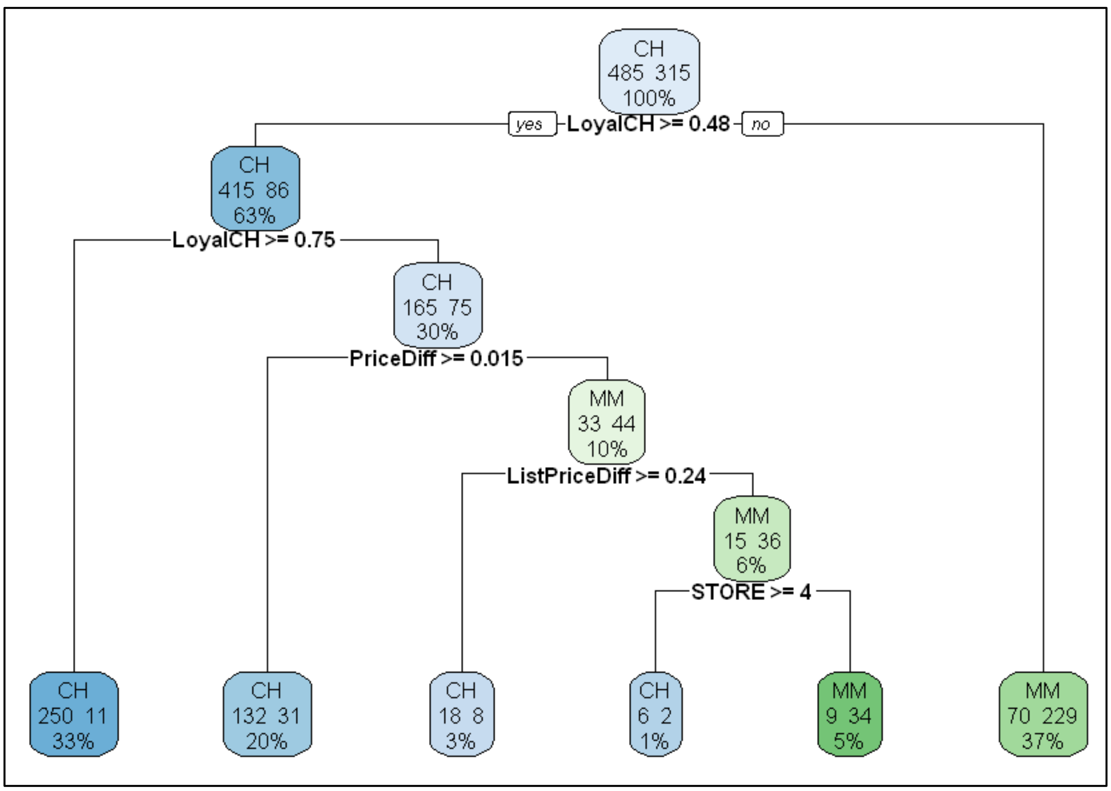
f) Apply the cv.tree() function to the training set in order to determine the optimal tree size. What is the optimal cp value?

```
> # q2
> # a
> data("OJ", package = "ISLR")
> head(OJ)
  Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH SpecialMM  LoyalCH
1       CH            237       1    1.75    1.99   0.00    0.0         0         0 0.500000
2       CH            239       1    1.75    1.99   0.00    0.3         0         1 0.600000
3       CH            245       1    1.86    2.09   0.17    0.0         0         0 0.680000
4       MM            227       1    1.69    1.69   0.00    0.0         0         0 0.400000
5       CH            228       7    1.69    1.69   0.00    0.0         0         0 0.956535
6       CH            230       7    1.69    1.99   0.00    0.0         0         1 0.965228
  SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM PctDiscCH ListPriceDiff STORE
1        1.99        1.75      0.24     No  0.000000  0.000000          0.24     1
2        1.69        1.75     -0.06     No  0.150754  0.000000          0.24     1
3        2.09        1.69      0.40     No  0.000000  0.091398          0.23     1
4        1.69        1.69      0.00     No  0.000000  0.000000          0.00     1
5        1.69        1.69      0.00    Yes  0.000000  0.000000          0.00     0
6        1.99        1.69      0.30    Yes  0.000000  0.000000          0.30     0
> dim(OJ)
[1] 1070   18
> # Variable Names:
> names(OJ)
 [1] "Purchase"       "WeekofPurchase" "StoreID"        "PriceCH"        "PriceMM"
 [6] "DiscCH"         "DiscMM"         "SpecialCH"      "SpecialMM"      "LoyalCH"
[11] "SalePriceMM"    "SalePriceCH"    "PriceDiff"      "Store7"         "PctDiscMM"
[16] "PctDiscCH"      "ListPriceDiff"  "STORE"
> attach(OJ)
>
>
> # b
> install.packages("caret")
> library(caret)
> set.seed(037831852)
> train_index <- sample(1:nrow(OJ), 800, replace = FALSE)
> train_set <- OJ[train_index, ]
> test_set <- OJ[-train_index, ]
> dim(train_set)
[1] 800  18
```

```
> dim(test_set)
[1] 270  18
>
>
> # c
> install.packages("rpart")
> library(rpart)
>
> tree_model <- rpart(Purchase ~ ., data = train_set, method = "class")
> summary(tree_model)
Call:
rpart(formula = Purchase ~ ., data = train_set, method = "class")
  n= 800

          CP nsplit rel error    xerror       xstd
1 0.50476190      0 1.0000000 1.0000000 0.04387030
2 0.01746032      1 0.4952381 0.5269841 0.03641188
3 0.01269841      4 0.4285714 0.4920635 0.03548871
4 0.01000000      5 0.4158730 0.5047619 0.03583208

Variable importance
       LoyalCH      PriceDiff    SalePriceMM      PctDiscMM        PriceMM         DiscMM 
            60              7              6              5              5              5 
 ListPriceDiff        StoreID WeekofPurchase        PriceCH          STORE 
             4              3              2              2              2 

Node number 1: 800 observations,    complexity param=0.5047619
  predicted class=CH  expected loss=0.39375  P(node) =1
    class counts:   485   315
   probabilities: 0.606 0.394 
  left son=2 (501 obs) right son=3 (299 obs)
  Primary splits:
      LoyalCH     < 0.48285   to the right, improve=132.23840, (0 missing)
      StoreID     < 3.5       to the right, improve= 33.91257, (0 missing)
      PriceDiff   < 0.31      to the right, improve= 22.10111, (0 missing)
      SalePriceMM < 1.84      to the right, improve= 19.62293, (0 missing)
      DiscCH      < 0.165     to the right, improve= 16.44038, (0 missing)
  Surrogate splits:
      PriceMM        < 1.89      to the right, agree=0.639, adj=0.033, (0 split)
      StoreID        < 3.5       to the right, agree=0.637, adj=0.030, (0 split)
      DiscMM         < 0.57      to the left,  agree=0.634, adj=0.020, (0 split)
      PctDiscMM      < 0.264375  to the left,  agree=0.634, adj=0.020, (0 split)
      WeekofPurchase < 227.5     to the right, agree=0.632, adj=0.017, (0 split)

Node number 2: 501 observations,    complexity param=0.01746032
  predicted class=CH  expected loss=0.1716567  P(node) =0.62625
    class counts:   415    86
   probabilities: 0.828 0.172 
  left son=4 (261 obs) right son=5 (240 obs)
  Primary splits:
      LoyalCH     < 0.7535455 to the right, improve=18.277250, (0 missing)
      PriceDiff   < 0.015     to the right, improve=13.650540, (0 missing)
      SalePriceMM < 1.84      to the right, improve=11.703510, (0 missing)
      PctDiscMM   < 0.1961965 to the left,  improve= 7.788811, (0 missing)
      DiscMM      < 0.47      to the left,  improve= 7.008025, (0 missing)
  Surrogate splits:
      PriceCH        < 1.825     to the right, agree=0.607, adj=0.179, (0 split)
      PriceMM        < 2.04      to the right, agree=0.601, adj=0.167, (0 split)
      SalePriceMM    < 2.04      to the right, agree=0.597, adj=0.158, (0 split)
      WeekofPurchase < 239.5     to the right, agree=0.595, adj=0.154, (0 split)
      StoreID        < 3.5       to the right, agree=0.581, adj=0.125, (0 split)

Node number 3: 299 observations
  predicted class=MM  expected loss=0.2341137  P(node) =0.37375
    class counts:    70   229
   probabilities: 0.234 0.766 
```

```
Node number 4: 261 observations
  predicted class=CH  expected loss=0.04214559  P(node) =0.32625
    class counts:   250    11
   probabilities: 0.958 0.042 

Node number 5: 240 observations,    complexity param=0.01746032
  predicted class=CH  expected loss=0.3125  P(node) =0.3
    class counts:   165    75
   probabilities: 0.688 0.313 
  left son=10 (163 obs) right son=11 (77 obs)
  Primary splits:
      PriceDiff     < 0.015     to the right, improve=15.202130, (0 missing)
      ListPriceDiff < 0.18      to the right, improve=13.225000, (0 missing)
      SalePriceMM   < 1.84      to the right, improve=11.814630, (0 missing)
      DiscMM        < 0.15      to the left,  improve= 6.054725, (0 missing)
      PctDiscMM     < 0.0729725 to the left,  improve= 6.054725, (0 missing)
  Surrogate splits:
      SalePriceMM   < 1.84      to the right, agree=0.950, adj=0.844, (0 split)
      PctDiscMM     < 0.1155095 to the left,  agree=0.887, adj=0.649, (0 split)
      DiscMM        < 0.27      to the left,  agree=0.875, adj=0.610, (0 split)
      ListPriceDiff < 0.18      to the right, agree=0.787, adj=0.338, (0 split)
      PriceMM       < 2.04      to the right, agree=0.779, adj=0.312, (0 split)

Node number 10: 163 observations
  predicted class=CH  expected loss=0.190184  P(node) =0.20375
    class counts:   132    31
   probabilities: 0.810 0.190 

Node number 11: 77 observations,    complexity param=0.01746032
  predicted class=MM  expected loss=0.4285714  P(node) =0.09625
    class counts:    33    44
   probabilities: 0.429 0.571 
  left son=22 (26 obs) right son=23 (51 obs)
  Primary splits:
      ListPriceDiff < 0.235     to the right, improve=5.460892, (0 missing)
      DiscMM        < 0.47      to the left,  improve=3.722482, (0 missing)
      PctDiscMM     < 0.227263  to the left,  improve=3.722482, (0 missing)
      StoreID       < 3.5       to the right, improve=3.238676, (0 missing)
      PriceCH       < 1.755     to the left,  improve=3.238676, (0 missing)
  Surrogate splits:
      PriceDiff   < -0.165    to the right, agree=0.753, adj=0.269, (0 split)
      PriceCH     < 1.755     to the left,  agree=0.740, adj=0.231, (0 split)
      SalePriceCH < 1.755     to the left,  agree=0.740, adj=0.231, (0 split)
      StoreID     < 5.5       to the right, agree=0.714, adj=0.154, (0 split)
      SpecialCH   < 0.5       to the right, agree=0.714, adj=0.154, (0 split)

Node number 22: 26 observations
  predicted class=CH  expected loss=0.3076923  P(node) =0.0325
    class counts:    18     8
   probabilities: 0.692 0.308 

Node number 23: 51 observations,    complexity param=0.01269841
  predicted class=MM  expected loss=0.2941176  P(node) =0.06375
    class counts:    15    36
   probabilities: 0.294 0.706 
  left son=46 (8 obs) right son=47 (43 obs)
  Primary splits:
      STORE         < 3.5       to the right, improve=3.943912, (0 missing)
      ListPriceDiff < 0.115     to the left,  improve=3.595725, (0 missing)
      StoreID       < 2.5       to the right, improve=3.388778, (0 missing)
      PctDiscMM     < 0.1961965 to the left,  improve=2.160023, (0 missing)
      LoyalCH       < 0.51      to the right, improve=2.152080, (0 missing)

Node number 46: 8 observations
  predicted class=CH  expected loss=0.25  P(node) =0.01
    class counts:     6     2
   probabilities: 0.750 0.250 
```

```
Node number 47: 43 observations
  predicted class=MM  expected loss=0.2093023  P(node) =0.05375
    class counts:     9    34
   probabilities: 0.209 0.791

> cat("Root node: error rate = 0.39375", "\n",
+     "Most important variable: LoyalCH", "\n",
+     "Best first split: LoyalCH < 0.48285")
Root node: error rate = 0.39375
 Most important variable: LoyalCH
 Best first split: LoyalCH < 0.48285
> sum(tree_model$frame$var == "<leaf>")
[1] 6
> cat("There are 6 leaf/terminal nodes in the Tree")
There are 6 leaf/terminal nodes in the Tree
>
>
> # d
> install.packages("rpart")
Error in install.packages : Updating loaded packages
> library(rpart)
> install.packages("rpart.plot")
> library(rpart.plot)
> # ?rpart.plot
> rpart.plot(tree_model, extra = 101, cex = 0.65)
> cat("Inference: LoyalCH is the strongest predictor")
Inference: LoyalCH is the strongest predictor
```




```
>
> # e
> install.packages("rpart")
Error in install.packages : Updating loaded packages
> library(rpart)
> install.packages("caret")
> library(caret)
```

```
> predictions <- predict(tree_model, test_set, type = "class")
> conf_matrix <- table(Predicted = predictions, Actual = test_set$Purchase)
> conf_matrix
         Actual
Predicted  CH  MM
       CH 140  14
       MM  28  88
> test_error_rate <- 1 - sum(diag(conf_matrix)) / sum(conf_matrix)
> # or
> # (28+14)/(140+14+28+88)
> cat("Test Error rate: ", test_error_rate)
Test Error rate:  0.1555556
>
>
> # f
> install.packages("tree")
> library(tree)
> set.seed(037831852)
> tree_model_tree <- tree(Purchase ~ ., data = train_set)
> cv_results <- cv.tree(tree_model_tree, FUN = prune.tree)
> optimal_size <- cv_results$size[which.min(cv_results$dev)]
> a = list(cv_results = cv_results, optimal_size = optimal_size)
> cat("Optimal Size:", a$optimal_size)
Optimal Size: 6
```