Project: Modeling with Classification Trees

Ali Ladha

Introduction

Data analysis should be reproducible, meaning: every step taken to manipulate, clean, transform, summarize, visualize or model data should be documented exactly so that results can be replicated. RMarkdown is a tool—or, specifically, a document type—for doing reproducible data science by keeping the code for a project together with the written analysis and interpretation.

This is an RMarkdown template that you can use for calculating answers to the project quiz questions for this module. You will also knit this document to HTML (or Word) and submit it for the File Upload assignment.

RMarkdown uses a very simple markup language. For example, rather than interacting with a menu to format the text, as in MS Word, you use simple code outside of the code chunks:

- A dash and a space at the beginning of a line (as here) creates a bullet point for use in a list.
- A number with a period at the beginning of a line creates a numbered list.
- Hashtags create headings (#), subheadings (##) or sub-subheadings (###).
- Emphasis can be added with asterisks like this: *italics* and **bolding.**

Notes on compiling this document

- Change the information in the yaml header above: title and author.
- Make sure the output argument is set correctly. It should read: output: html_document or output: word_document.
- Once you are finished writing the code necessary to answer the questions in the quiz, clear your environment by clicking on the broom icon in the environment pane (upper right quadrant).
- Run each code chunk individually (click the green arrow icon in the upper right of the chunk). Start at the top of this document and proceed sequentially to the bottom. Fix any code errors you find.
- Once your code is error-free, click “knit” in the menu above. Your document should compile to HTML, if the output is set to “html_document” (or to word if the output is set to “word_document”).

In the code chunk above (entitled “setup”) echo is set to TRUE. This means that the code in your chunks will be displayed, along with the results, in your compiled document.

Load and Transform Data

Below is code to clean and prepare the dataset for modeling. Before running that code, follow these preparatory steps:

1. After downloading the RMarkdown template and the dataset for the assignment from Canvas, make sure to copy or move these files from your downloads folder to a folder dedicated to this class–say, MKTG-6487.
2. You need to define that folder as your “working directory.” To do so, navigate to that folder using the files tab in the lower right quadrant in RStudio. (You should see your files you moved into this folder in the

previous step.) Click the “More” button in the menu under the Files tab and select “Set As Working Directory.”

Once the files are in the right location on your computer then run this code to clean and format the data:

```
# You must run this code to format the dataset properly!

advise_invest <- read_csv("adviseinvest.csv")  |>           # Download data and save it
(via assignment operator)
  select(-product) |>                                       # Remove the product column
  filter(income > 0,                                         # Filter out mistaken data
         num_accts < 5) |>
  mutate(answered = ifelse(answered==0, "no","yes"),          # Turn answered into yes/n
o
         answered = factor(answered,                         # Turn answered into facto
r
                           levels  = c("no", "yes")),        # Specify factor levels
         female = factor(female),                           # Make other binary and ca
tegorical
# variables into factors
         job = factor(job),
         rent = factor(rent),
         own_res = factor(own_res),
         new_car = factor(new_car),
         mobile = factor(mobile),
         chk_acct = factor(chk_acct),
         sav_acct = factor(sav_acct))
```

```
## Rows: 29502 Columns: 14
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (14): answered, income, female, age, job, num_dependents, rent, own_res,...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Questions

Use the code chunks below to write code that will enable you to answer the questions in the project quiz.

Some of the questions do not require writing code and have been omitted from this template.

Q2.

```
mean(advise_invest$answered == "yes")
```

```
## [1] 0.5465948
```

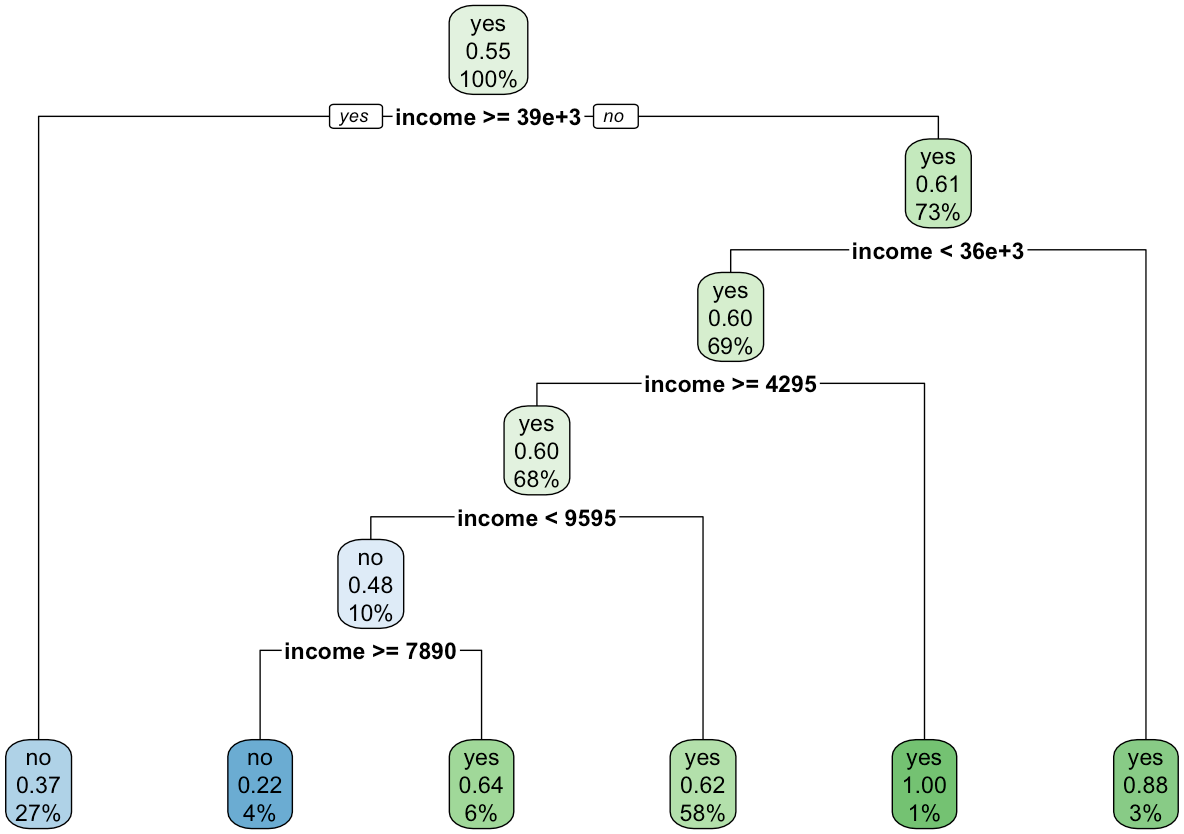
```
# OR ifelse(advise_invest$answered == "yes", 1,0) %>% mean()
# OR advise_invest %>% count(answered)
  #16124/29499
```

Q3.

```
income_model <- rpart(formula = answered ~ income, data = advise_invest)
income_model
```

```
## n= 29499
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 29499 13375 yes (0.4534052 0.5465948)
##    2) income>=39135 8063  2944 no (0.6348754 0.3651246) *
##    3) income< 39135 21436  8256 yes (0.3851465 0.6148535)
##      6) income< 35625 20412  8128 yes (0.3981971 0.6018029)
##       12) income>=4295 20156  8128 yes (0.4032546 0.5967454)
##         24) income< 9595 2944  1408 no (0.5217391 0.4782609)
##           48) income>=7890 1152   256 no (0.7777778 0.2222222) *
##           49) income< 7890 1792   640 yes (0.3571429 0.6428571) *
##         25) income>=9595 17212  6592 yes (0.3829886 0.6170114) *
##       13) income< 4295 256     0 yes (0.0000000 1.0000000) *
##      7) income>=35625 1024   128 yes (0.1250000 0.8750000) *
```

```
rpart.plot(income_model)
```




```
(predict(income_model, type = "class") == advise_invest$answered) %>% mean()
```

```
## [1] 0.6420218
```

```
  #OR you can use rpart and manually calculate it by taking observations and subtracting
them by lossing and dividing that
  # by total sum of observations:
  # (8063-2944) + (1152-256) + (1792 - 640) + (17212-6592) + 256 + (1024-128) / 29499
```

Q4.

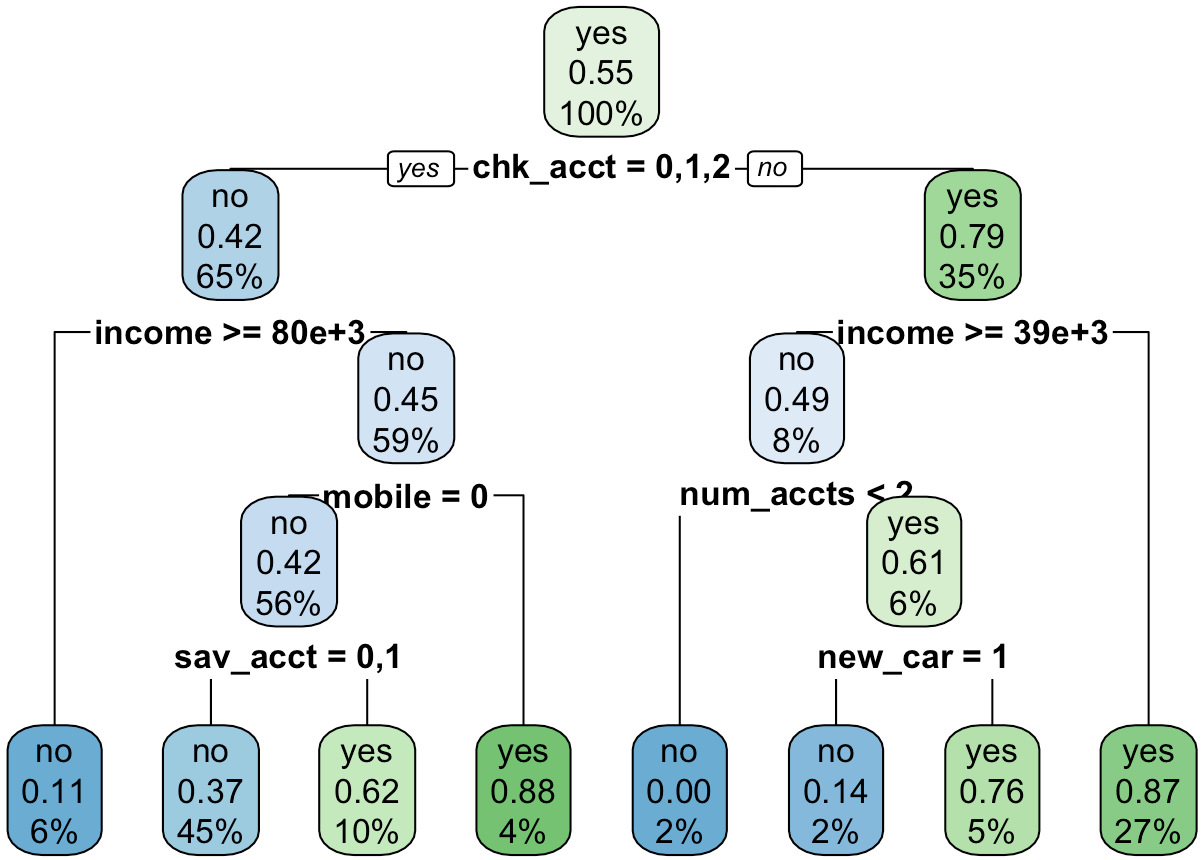
The classification tree algorithm automatically splits data to maximize information gain (while minimizing entropy). This ensures a good result as the data is split more homogeneously. The focus of the algorithm is to have a higher information gain, and have a lower entropy. An Entrophy of 0 would mean that the leaf nodes would all be either answered or unanswered. Whereas an Entropy of 1 means they are equally split.

Q5.

```
tree_model <- rpart(formula = answered ~., data = advise_invest, maxdepth = 4)
tree_model
```

```
## n= 29499
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 29499 13375 yes (0.4534052 0.5465948)
##    2) chk_acct=0,1,2 19199  8000 no (0.5833116 0.4166884)
##      4) income>=79840 1728   192 no (0.8888889 0.1111111) *
##      5) income< 79840 17471  7808 no (0.5530880 0.4469120)
##       10) mobile=0 16383  6848 no (0.5820057 0.4179943)
##         20) sav_acct=0,1 13311  4928 no (0.6297799 0.3702201) *
##         21) sav_acct=2,3,4 3072  1152 yes (0.3750000 0.6250000) *
##       11) mobile=1 1088   128 yes (0.1176471 0.8823529) *
##    3) chk_acct=3 10300  2176 yes (0.2112621 0.7887379)
##      6) income>=38910 2240  1088 no (0.5142857 0.4857143)
##       12) num_accts< 1.5 448     0 no (1.0000000 0.0000000) *
##       13) num_accts>=1.5 1792   704 yes (0.3928571 0.6071429)
##         26) new_car=1 448    64 no (0.8571429 0.1428571) *
##         27) new_car=0 1344   320 yes (0.2380952 0.7619048) *
##      7) income< 38910 8060  1024 yes (0.1270471 0.8729529) *
```

```
rpart.plot(x = tree_model, tweak = 1.2, roundint = T)
```



Based on the plot the most important predictions is the: Chk_acct, income, and mobile variables.

Q6.

```
(predict(income_model, type = "class") ==  (advise_invest$answered)) %>% mean()
```

```
## [1] 0.6420218
```

```
(predict(tree_model, type = "class") == (advise_invest$answered)) %>% mean()
```

```
## [1] 0.7353131
```

The tree_model is better since it has a higher accuracy rate.