IBA - Project: Modeling with Classification Trees

Prabhudatta Mohapatra

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 a
ssignment 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/no
         answered = factor(answered,                         # Turn answered into factor
                           levels  = c("no", "yes")),        # Specify factor levels
         female = factor(female),                           # Make other binary and categorica
l
# 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.

Q1.

Classification model accuracy is defined as the proportion of correct classifications done by the classification model. It ranges between 0 and 1. It is calculated as ((number of observations on left node - number of incorrect classification on left node) +

(number of observations on right node - number of incorrect classification on right node))/ total number of observations.

A regression model is used when the target variable is continuous while a classification model is used when the target variable is

binary or multi-class. Regression model predicts continuous target and we cannot classify it into a class. So, we cannot calculate

proportion of correct classifications; hence, accuracy cannot be used to evaluate the performance of a regression model.

Q2.

```
# Majority Class classifier accuracy
round(prop.table(table(advise_invest$answered)),3)
```

```
##
##    no   yes
## 0.453 0.547
```

Q3.

```
# Income Model
income_model <- rpart(formula = answered ~ income,
                      data = advise_invest)
# Income Model Accuracy
(predict(object = income_model, type = "class") == advise_invest$answered) %>% mean %>% round(3)
```

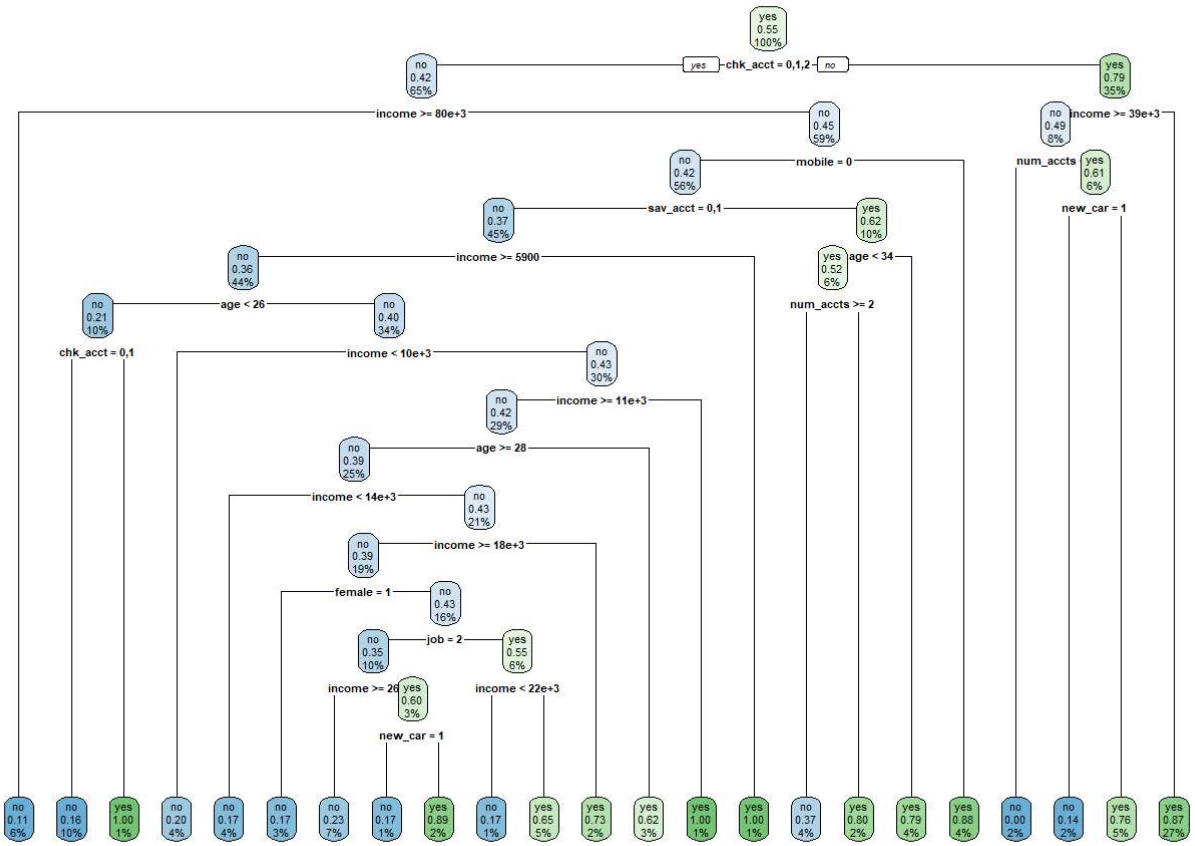
```
## [1] 0.642
```

Q4.

Entropy is a measure of disorder that can be applied to a set to indicate how mixed or impure it is with respect to a property of interest and information gain is a measure of change in entropy due to any amount of new information being added. Entropy thus inversely measures the purity of a node, as purity goes down entropy goes up. Information gain measures the change in node purity resulting from a split. So the primary goal of a split such as income >= 39135 and < 39135 to find splits that makes node with least possible entropy or disorder. After the split the two nodes should together have higher purity or lower entropy than the original node. Information gain evaluate possible splits to identify the one best split that decreases the entropy most (increases children node purity) and this is how tree algorithm knows where to split the data

Q5.

```
# Model using all the predictors
tree_model<- rpart(formula = answered ~ .,
                       data = advise_invest)
rpart.plot(tree_model, tweak = 1.5,roundint = T, compress = T)
```




```
#Important Predictor
tree_model$variable.importance
```

```
##       chk_acct         income      num_accts            age       new_car
##    2184.231096    2182.002993     570.658073     524.361531     497.879365
##         mobile       sav_acct            job         female       own_res
##     439.988872     397.966598     166.123982     100.974137      86.919804
##           rent num_dependents
##      62.480726       5.731206
```

```
##Checking Account (chk_acct)
##Income
##Number of Accounts (num_accts)
```

Q6.

```
# Tree Model Accuracy
(predict(object = tree_model, type = "class") == advise_invest$answered) %>% mean %>% round(3)
```

```
## [1] 0.82
```

Q7.

This classification model will be helpful to understand if a potential customer will answer a scheduled call or not. Also, the three important factors that determine if a potential customer will answer are what type of checking account the customer has, what is the income level of the customer, and how many accounts the customer has. Model-generated probabilities can be used to "score" or rank customers based on their likelihood to answer the call. This will help the representatives to prioritize their scheduled calls and accordingly plan the engagement during peak hours as well as through out the day. If this model works well in real-time then based on the outcome we can predcit how many calls a sales representative should make in an hour to accomodate maximum number of calls and improve representative utilization.