IBA_Project: Model Evaluation and Deployment

Prabhudatta Mohapatra

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 data set for modeling. Before running that code, follow these preparatory steps:

1. Download the RMarkdown template and the data sets for the assignment from Canvas.

2. Copy or move these files from your downloads folder to a folder dedicated to this class--say, MKTG-6487.

3. You need to define this 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 data set properly!

advise_invest <- read_csv("adviseinvest.csv")  %>%            # Download data
  select(-product) %>%                                        # Remove the product column
  filter(income > 0,                                          # Filter out mistaken data
         num_accts < 5) %>%
  mutate(answered = factor(ifelse(answered==0, "no","yes"),   # Turn answered into yes/no factor
                           levels  = c("no", "yes")),
         female = factor(female),                             # Make categorical 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))
```

And here is code to load the data set of prospective customers from your working directory. Note that in order to use this data set for prediction, the variables need to be formatted exactly the same as in the data used to fit the model. It does not include a target variable because the event of answering or not answering has not happened yet for scheduled customers.

```
prospective <- read_csv("customer_data.csv") %>%
  mutate(female = factor(female),
         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))
```

Questions

One of the simplifying assumptions we will make in this project is that all the customers who answer the phone will purchase a product. (This assumption is actually verified by the data.) To model "answered" in this case is therefore equivalent to modeling "purchased."

There are costs and benefits in this case. We will assume that customers purchase a product for $100 dollars. This was the average cost of AdviseInvest products, according to the Director of Sales. Also, as we learned in the interview, the agent time to make the sale is worth $25. Profit would therefore be $75 dollars for an answered call and a purchase. In sum:

Benefit: True positive. The customer is predicted to answer, does answer, and purchases a product for $100 for a profit of 100 - 25 = $75.

Cost: False positive. The customer is predicted to answer, but does not answer, so there is a loss of $25. (We assume the agent cannot schedule another call at the last minute, or spends the entire time slot trying to make the call.)

For this exercise, we propose that customers who are not predicted to answer will not be called, so there would be no benefits and no costs for them.

However, this proposal is for illustration only. Below you will be asked to come up with a final recommendation for the Director of Sales, and you should feel free to craft a solution---whatever that might be---that fits the details of the case.

One thing to keep in mind for this final phase of the project is that a predictive model is always developed using historical data. The end goal, however, is to predict the future occurrence of the event that has been modeled. In this exercise, you will practice using data on new customers---that is, customers who have not yet been called---to predict whether they will answer. How you use these predictions in solving the business problem is up to you.

Q1.

```
#tree model (answered)
tree_model <- rpart(formula = answered ~ .,
                      data = advise_invest)
#confusion matrix
table(predicted = predict(tree_model, type = "class"),
      observed = advise_invest$answered)
```

```
##          observed
## predicted    no   yes
##       no  10367  2304
##       yes  3008 13820
```

Q2

```
#Profit
(13820*75) - (3008*25)
```

```
## [1] 961300
```

Q3

```
# tree model with class decision threshold >= 0.3
table(predicted = ifelse(predict(tree_model, type = "prob")[,2] >= 0.3, "Yes", "No"),
      observed = advise_invest$answered)
```

```
##          observed
## predicted    no   yes
##       No   9599  1856
##       Yes  3776 14268
```

```
#Profit
(14268*75) - (3776*25)
```

```
## [1] 975700
```

Q4

```
#Getting predictied probablities
predictions <- prospective %>%
  select(customer_id) %>%
  mutate(ans_prob = predict(tree_model,
                            newdata = prospective,
                            type = "prob")[,2])

head(predictions)
```

```
## # A tibble: 6 x 2
##   customer_id ans_prob
##   <chr>          <dbl>
## 1 H1597          0.226
## 2 P1446          0.226
## 3 E1492          0.167
## 4 W5143          0.727
## 5 W4927          0.762
## 6 M6161          0.873
```

```
# Filtering the customers based on assigned class decision threshold
ans_customer_list <- predictions %>%
  filter(ans_prob >= .3) %>%
  arrange(desc(ans_prob))

glimpse(ans_customer_list)
```

```
## Rows: 624
## Columns: 2
## $ customer_id <chr> "M8502", "V4331", "S3197", "S6624", "C1491", "W3586", "Y63~
## $ ans_prob    <dbl> 1.0000000, 1.0000000, 1.0000000, 1.0000000, 1.0000000, 1.0~
```

Q5

Using the decision tree model, cost-benefit matrix, and optimized class decision threshold (0.3) a list of 624 customers has been generated who will potentially answer the scheduled call.

The call center representatives should focus on these customers and plan their day to prioritize the calls with these 624 customers as they will answer the call and buy AdviseInvest products.

Further, the tree model should be used continuously to identify customers who will answer the call. Also, new historical data should be added to the decision tree model to improve performance. It will help reduce misclassification (particularly false positives), which adds the cost.

In the future, we will follow up with additional analysis to determine how many calls each sales representative should make in an hour having the knowledge that the prediction model for customers answering the call is working well.