Class 14 Demo: Next-Product-to-Buy Modeling with Logistic Regression

Load packages and read in data

```
### Load packages:
library(janitor)
#library(haven)
#library(readxl)
#library(knitr)
#library(psych)
#library(statar)
library(tidyverse)
#library(mktg482)
#library(readstata13)
#library(printr)

# Clear environment of datasets
rm(list=ls())

# Read in and transform data
set.seed(1004)
load("../../Data/bbb_nptb_email.RData")
#str(bbb.nptb)
bbb.nptb <- bbb.nptb %>% mutate(buyer=factor(buyer), gender=factor(gender))
bbb.nptb.toscore <- bbb.nptb.toscore %>% mutate(buyer=factor(buyer), gender=factor(gender))
```

This notebook shows how to do Next-Product-To-Buy modeling using a test performed by Bookbinders. Stan Lawton sent:

- 10,000 randomly selected customers an offer containing "The Art History Of Florence" as representative of the "Art" category.
- 10,000 randomly selected customers an offer containing "Vegetarian Cooking for Everyone" as representative of the "Cook" category.
- 10,000 randomly selected customers an offer containing "Painting Like a Pro" as representative of the "DIY" category.

The main dataset `bbb.nptb` contains the results for these 30,000 customers (we will return to `bbb.nptb.toscore` at the end this demo). Our task is to use these results to find the best book to offer each customer. The dataset contains a `training` variable we can use to create a train/test split.

Part I: Estimating purchase probabilities

The idea of customization is that different offers may work Hence, we want to calculate for each customer the probability that they would buy after receiving either an art book, a cookbook, or a DIY book.

But, each customer received only one book. So, for each book, we will use the customers who received that book to fit a model which we then use to predict for all customers.

First, let's split the data into training and test sets.

```
bbb.nptb.train <- bbb.nptb %>% filter(training==1)
bbb.nptb.test <- bbb.nptb %>% filter(training==0)
```

Now, let's use the data for customers who received an art book to create a model and then use that model to predict the probability of responding if sent an art book for *all* customers. To do this, we must estimate the logistic regression only on customers who received `offer=="Art"` (in training) and then predict for *all* customers regardless of what offer they received (in test).

```
lr.art <- glm(buyer ~ gender + last + total + child + youth + cook + do_it + reference + art + geog,
              family=binomial, data=bbb.nptb.train %>% filter(offer=="Art"))
bbb.nptb.test <- bbb.nptb.test %>%
  mutate(pr.art = predict(lr.art, newdata=bbb.nptb.test, type="response"))
```

We now do the same for the cook and DIY offers.

```
lr.cook <- glm(buyer ~ gender + last + total + child + youth + cook + do_it + reference + art + geog,
              family=binomial, data=bbb.nptb.train %>% filter(offer=="Cook"))
bbb.nptb.test <- bbb.nptb.test %>%
  mutate(pr.cook = predict(lr.cook, newdata=bbb.nptb.test, type="response"))

lr.diy <- glm(buyer ~ gender + last + total + child + youth + cook + do_it + reference + art + geog,
              family=binomial, data=bbb.nptb.train %>% filter(offer=="DIY"))
bbb.nptb.test <- bbb.nptb.test %>%
  mutate(pr.diy = predict(lr.diy, newdata=bbb.nptb.test, type="response"))
```

We now have predicted purchase probabilities for each book for each customer in the test dataset. We now want to figure out what book offer yields the highest predicted purchase probability for each customer.

We start by creating a new variable `pr.max` that contains the highest predicted purchase probability for each customer. This will allow us to customize the "Next Product to Buy" offer to each customer. Specifically, if `pr.max==pr.art` we should offer the customer the art book; if `pr.max==pr.cook` we should offer the customer the cookbook; and if `pr.max==pr.diy` we should offer the customer the DIY book.

```
bbb.nptb.test <- bbb.nptb.test %>%
  mutate(pr.max = pmax(pr.art, pr.cook, pr.diy),
         mail.offer = case_when(
             pr.art == pr.max ~ "Art",
             pr.cook == pr.max ~ "Cook",
             pr.diy == pr.max ~ "DIY"))
```

Now, let's see what number of each book we should send:

```
bbb.nptb.test %>%
  tabyl(mail.offer)
```

```
 mail.offer    n   percent
        Art 2268 0.2520000
       Cook 3323 0.3692222
        DIY 3409 0.3787778
```

Part II: How much better have we done?

To evaluate the improvement in what we have done, we need a "no targeting" benchmark. In an NPTB case, this benchmark is the purchase probability if you send out the *same* book to everyone (i.e., no customization). As it turns out, we already know the purchase probability for each of the three books, namely `pr.art`, `pr.cook`, and `pr.diy`.

In contrast, if we target using the book with the highest purchase probability for each consumer (i.e., customization), then the expected purchase probability is simply the maximum of the three for each customer. We already calculated this as `pr.max`!

So, we can easily compare the different options:

```
bbb.nptb.test %>%
  summarise_at(vars(pr.art, pr.cook, pr.diy, pr.max), list(mean))
```

```
# A tibble: 1 x 4
  pr.art pr.cook pr.diy pr.max
   <dbl>   <dbl>  <dbl>  <dbl>
1 0.0892   0.132  0.107  0.224
```

```
# EQUIVALENTLY:
# bbb.nptb.test %>%
#   summarise(mean(pr.art),mean(pr.cook),mean(pr.diy),mean(pr.max))
```

The improvement in the purchase probability using targeting is tremendous.

Note: Previously we did not use the mean of the model predictions to forecast profit but what actually happened, that is, the mean of the `buyer` variable. We can still do that here for each of the three offers separately, but this is not possible for customization because we did not actually implement this; we must rely on the model!

```
# Compare to what we did previously but which we cannot do for customization option:
bbb.nptb.test %>%
  group_by(offer) %>%
  summarize(purchases=sum(buyer==1), customers=n(), rate=mean(buyer==1))
```

```
# A tibble: 3 x 4
  offer purchases customers   rate
  <chr>     <int>     <int>  <dbl>
1 Art         276      3039 0.0908
2 Cook        361      2962 0.122
3 DIY         349      2999 0.116
```

Part III: What if different offers yield different profit?

So far we have picked offers for each customer according to their predicted purchase probabilities. However, that is not the right criterion if offers differ in how profitable they are. In this case, let's assume the following:

- The profit from selling the "Art History of Florence" is $6.
- The profit from selling the "Vegetarian Cooking for Everyone" is $7.
- The profit from selling the "Painting Like a Pro" is $4.

Now, we have to calculate the expected profits for each book for each customer, which is simply each one of the three profits times the respective predicted purchase probability. I use the prefix "`ep`" for these variables, short for "expected profit."

```
bbb.nptb.test <- bbb.nptb.test %>%
  mutate(ep.art=pr.art*6, ep.cook=pr.cook*7, ep.diy=pr.diy*4)
```

The rest of the calculation is exactly as above.

```
bbb.nptb.test <- bbb.nptb.test %>%
  mutate(ep.max = pmax(ep.art, ep.diy, ep.cook),
         mail.offer.ep = case_when(
```

```
            ep.art == ep.max ~ "Art",
            ep.cook == ep.max ~ "Cook",
            ep.diy == ep.max ~ "DIY"))
```

Now, let's see what number of each book we should send and compare to what happened when we targeted only by probability:

```
bbb.nptb.test %>%
  tabyl(mail.offer)
```

```
 mail.offer    n   percent
        Art 2268 0.2520000
       Cook 3323 0.3692222
        DIY 3409 0.3787778
```

```
bbb.nptb.test %>%
  tabyl(mail.offer.ep)
```

```
 mail.offer.ep    n   percent
           Art 2312 0.2568889
          Cook 4024 0.4471111
           DIY 2664 0.2960000
```

```
bbb.nptb.test %>%
  tabyl(mail.offer, mail.offer.ep) %>%
  adorn_title()
```

```
            mail.offer.ep
 mail.offer           Art Cook  DIY
        Art          1882  386    0
       Cook             0 3323    0
        DIY           430  315 2664
```

This distributions are quite different.

How much better have we done in terms of expected profit per customer as compared to the no customization benchmark? As above, if we target the book with the highest expected profit for each consumer, then the overall expected profit should be the maximum of the three for each customer. We already calculated this in `ep.max`. So, we can easily compare the different options:

```
bbb.nptb.test %>%
  summarise_at(vars(ep.art, ep.cook, ep.diy, ep.max), list(mean))
```

```
# A tibble: 1 x 4
  ep.art ep.cook ep.diy ep.max
   <dbl>   <dbl>  <dbl>  <dbl>
1  0.535   0.926  0.427   1.28
```

Part IV: How do we use this outside the test?

The dataset `bbb.nptb.toscore` contains an additional 10,000 customers that were not part of the test. For these customers, neither the offer nor the buyer variable is specified. We can easily score this new data by rerunning the `predict` command on it:

```
bbb.nptb.toscore <- bbb.nptb.toscore %>%
  mutate(pr.art = predict(lr.art, newdata=bbb.nptb.toscore, type="response"),
```

```
         pr.cook = predict(lr.cook, newdata=bbb.nptb.toscore, type="response"),
         pr.diy = predict(lr.diy, newdata=bbb.nptb.toscore, type="response"))
```

We can now calculate which book yields the highest expected profit using the same calculations as before. Note that `mutate()` can immediately reuse a variable that we just created.

```
bbb.nptb.toscore <- bbb.nptb.toscore %>%
  mutate(ep.art = pr.art*6,
         ep.cook= pr.cook*7,
         ep.diy = pr.diy*4,
         ep.max = pmax(ep.art, ep.cook, ep.diy),
         mail.offer.ep = case_when(
             ep.art == ep.max ~ "Art",
             ep.cook == ep.max ~ "Cook",
             ep.diy == ep.max ~ "DIY"))
```

Now we know which book to send which customer, information that can go directly into our marketing automation systems.

```
bbb.nptb.toscore %>%
  select(acctnum, mail.offer.ep)
```

```
# A tibble: 10,000 x 2
   acctnum mail.offer.ep
     <dbl> <chr>
 1  440943 DIY
 2  335005 Cook
 3  431568 Cook
 4  240926 DIY
 5  319951 Cook
 6  290327 Art
 7  211456 DIY
 8  271586 DIY
 9   66717 Cook
10  426428 Art
# ... with 9,990 more rows
```