Code

- [Show All Code]
- [Hide All Code]
-
- [Download Rmd]

Targeted Optimal Bidding

In this notebook we see how we can model bidding data from a B2B retailer to a) figure out what determines the probability of the customer accepting a bid and b) determine the optimal price to quote for a given customer for a given bid. We will use logistic regression followed by optimization for this task.

We begin by reading the data from the file RawShort1.csv in our working directory.

```
wd<-getwd()
```

```
bd <- read.csv(file.path(wd,"RawShort1.csv"))
```

```
summary(bd)
```

```
    Customer          Order             Time             Quantity          PricePerLb        CostPerLb     
 Min.   :   1.0   Min.   :0.0000   Min.   : 0.07143   Min.   : 0.00069   Min.   : 0.289   Min.   :0.2094  
 1st Qu.: 435.0   1st Qu.:0.0000   1st Qu.: 0.42857   1st Qu.: 0.09494   1st Qu.: 2.377   1st Qu.:1.7008  
 Median : 992.0   Median :0.0000   Median : 1.71429   Median : 0.23345   Median : 2.754   Median :1.8094  
 Mean   : 921.9   Mean   :0.4122   Mean   : 4.45134   Mean   : 0.64477   Mean   : 3.149   Mean   :1.9618  
 3rd Qu.:1375.0   3rd Qu.:1.0000   3rd Qu.: 5.00000   3rd Qu.: 0.62200   3rd Qu.: 3.502   3rd Qu.:2.0300  
 Max.   :1818.0   Max.   :1.0000   Max.   :97.28571   Max.   :14.65804   Max.   :10.000   Max.   :7.7700  
    LagPrice     
 Min.   : 0.289  
 1st Qu.: 2.383  
 Median : 2.778  
 Mean   : 3.165  
 3rd Qu.: 3.521  
 Max.   :10.000  
```

Logistic Regression

We can perform a logistic regression and summarize it. We begin by running a logistic regression of the order (0/1) variable on the Time, Quantity and PricePerLb variables. We see below that all three coefficients are negative. This means that greater the time since the previous contact, the lower is the probability of our bid getting accepted, the larger is the quantity desired, the lower is the chance of acceptance, and the higher the price we quote, the lower is the chance of acceptance.

```
res1 <- glm(Order ~ Time+Quantity+PricePerLb, data=bd, family=binomial(link = "logit"))
summary(res1)
```

```

Call:
glm(formula = Order ~ Time + Quantity + PricePerLb, family = binomial(link = "logit"), 
    data = bd)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.3315  -1.0830  -0.8045   1.2208   3.1745  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.646405   0.055090   11.73   <2e-16 ***
Time        -0.031060   0.002587  -12.01   <2e-16 ***
Quantity    -0.485666   0.025220  -19.26   <2e-16 ***
PricePerLb  -0.193649   0.015139  -12.79   <2e-16 ***
---
Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1    1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 21532  on 15886  degrees of freedom
Residual deviance: 20702  on 15883  degrees of freedom
AIC: 20710

Number of Fisher Scoring iterations: 5
```

Predictions

Suppose we are interested in using the results to make predictions on future data. Let us take the first three observations of our data and store it in a dataframe df1.

```
df1<-bd[1:3,]
df1
```

We can use the results of our logistic regression (stored in the object res1) and make a prediction for the observation in df1.

```
predict.glm(res1, newdata=df1, type="response")
```

```
        1         2         3 
0.1487282 0.3674304 0.4717426 
```

We can write a function that computes the profits for a given price x and the values of the other variables. The function also takes as an argument, hold_obs, which contains the values of the other variables (CostPerLb and Quantity) that are needed for the profit computation. In the function below, we make sure we insert the new price x into the observation before computing the probabilities and the profit.

```
pf1<-function(x, hold_obs, result)
{
  df_hold <- hold_obs
  df_hold$PricePerLb <-x
  prob <- predict.glm(result, newdata=df_hold, type="response")
  profit <- (x-df_hold$CostPerLb)*prob*df_hold$Quantity
  profit
}
```

Let us compute the profit for the first observation of the data frame df1.

```
pf1(2, df1[1,], res1)
```

```
        1 
0.2205431 
```

We can optimize the profit by using the built-in optim function in R. We use a starting value of 4.0 for the optimal price, and store the optimization results in opt.

```
opt1<-optim(4.0, pf1, method="BFGS", control=list(fnscale=-1), hold_obs=df1[1,], result=res1)
```

The optimal price and the optimal profit can be extracted from the optimization result as below. Note that the optimal price of $7.09 appears too high to be reasonable, and thus we need to think critically about the model that we have specified. We now see how this model can be extended to include reference effects that may be operant in a given situation.

```
c(opt1$par, opt1$value)
```

```
[1] 7.092684 1.206603
```

Logistic regression with reference prices.

We create two variable, loss and gain, to capture reference price effects, as detailed in the handout. We then store this in the bd dataframe.

```
gain <- (bd$LagPrice - bd$PricePerLb)*(bd$LagPrice > bd$PricePerLb)
loss <- (bd$PricePerLb - bd$LagPrice)*(bd$PricePerLb > bd$LagPrice)
bd$gain <- gain
bd$loss <- loss
```

We now do a logistic regression that includes these reference effects and their interactions with the quantity variable.

```
res2 <- glm(Order ~ Time+Quantity+gain+loss+Quantity:gain+Quantity:loss, data=bd, family=binomial(link = "logit"))
summary(res2)
```

```

Call:
glm(formula = Order ~ Time + Quantity + gain + loss + Quantity:gain + 
    Quantity:loss, family = binomial(link = "logit"), data = bd)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
```

```
-1.5444  -1.0807  -0.7514   1.2060   3.1761

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.132228   0.027893   4.741 2.13e-06 ***
Time          -0.032481   0.002614 -12.426  < 2e-16 ***
Quantity      -0.502302   0.030791 -16.313  < 2e-16 ***
gain           0.106462   0.021236   5.013 5.35e-07 ***
loss          -0.323503   0.025130 -12.873  < 2e-16 ***
Quantity:gain  0.055454   0.018180   3.050  0.00229 **
Quantity:loss -0.138058   0.074370  -1.856  0.06340 .
---
Signif. codes:  0 □***□ 0.001 □**□ 0.01 □*□ 0.05 □.□ 0.1 □ □ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 21532  on 15886  degrees of freedom
Residual deviance: 20475  on 15880  degrees of freedom
AIC: 20489

Number of Fisher Scoring iterations: 5
```

Predictions

Suppose we are interested in using the results to make predictions on future data. Let us take the first two observations of our data and store it in a dataframe df2. Assume that these are from a holdout dataset.

```
df2<-bd[1:2,]
df2
```

We can use the results of our logistic regression (stored in the object res2) and make a prediction for the observation in df2.

```
predict.glm(res2, newdata=df2, type="response")
```

```
        1         2
0.1305808 0.2711572
```

We now write a profit function, that takes a price (x), a holdout observation, (hold_obs), and the result of a logistic regression (result) to compute the profit function. Note that for any price x, we need to compute the gain and loss variables before computing the profit. The profit is equal to (price-cost)*orderProbability*Quantity

```
pf2<-function(x, hold_obs, result)
{

  df1<-hold_obs

  df1$gain <- (df1$LagPrice - x)*(df1$LagPrice > x)
  df1$loss <- (x - df1$LagPrice)* (x > df1$LagPrice)

  prob <- predict.glm(result, newdata=df1, type="response")
  profit <- (x-df1$CostPerLb)*prob*df1$Quantity
  profit

}
```

We can compute the profit for the first observation of df2. Note that df2[1,] gives the first observation of the dataframe df.

```
pf2(2.5, df2[1,], res2)
```

```
        1
0.3310888
```

We can optimize the profit by using the built-in optim function in R. We use a starting value of 4.0 for the optimal price, and store the optimization results in opt.

```
opt2<-optim(4.0, pf2, method="BFGS", control=list(fnscale=-1), hold_obs=df2[1,], result=res2)
```

The optimal price and the optimal profit is given below. These values are very reasonable, compared to the previous ones that we obtained based on the simpler model.

```
c(opt2$par, opt2$value)
```

```
[1] 2.9259754 0.3547824
```