401_Hw7

Hongkai Lou

2024-11-19

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
customer <- read.csv('customer2.csv')
customer$logtarg <- log(customer$target + 1)
cat("\nCustomer Data Summary:\n")
```

```
##
## Customer Data Summary:
```

```r
summary(customer)
```

```
##        id               train            target           logtarg
##  Min.   :     957   Min.   :0.0000   Min.   :  0.000   Min.   :0.0000
##  1st Qu.: 4448960   1st Qu.:0.0000   1st Qu.:  0.000   1st Qu.:0.0000
##  Median : 8090750   Median :0.0000   Median :  0.000   Median :0.0000
##  Mean   : 8563488   Mean   :0.3308   Mean   :  3.241   Mean   :0.2529
##  3rd Qu.:13378724   3rd Qu.:1.0000   3rd Qu.:  0.000   3rd Qu.:0.0000
##  Max.   :16456238   Max.   :1.0000   Max.   :739.480   Max.   :6.6073
```

```r
orders_data <- read.csv('orders.csv')
orders_data <- orders_data %>%
  distinct(id, orddate, ordnum, .keep_all = TRUE)

orders_data <- orders_data %>%
  mutate(t = as.numeric(as.Date("2014-11-25", format="%Y-%m-%d") - as.Date(orddate, format="%d%b%Y")) /
summary(orders_data)
```

```
##        id              orddate              ordnum           category    
##  Min.   :     957   Length:102555      Min.   :   1018   Min.   : 1.00  
##  1st Qu.: 3887413   Class :character   1st Qu.: 365248   1st Qu.:14.00  
##  Median : 6109373   Mode  :character   Median : 690438   Median :20.00  
##  Mean   : 6678104                      Mean   : 669318   Mean   :32.64  
##  3rd Qu.: 8689962                      3rd Qu.: 982118   3rd Qu.:37.00  
##  Max.   :16456238                      Max.   :1256189   Max.   :99.00  
##       qty               price                t           
##  Min.   :  0.000   Min.   :   0.00   Min.   :0.002738  
##  1st Qu.:  1.000   1st Qu.:   6.95   1st Qu.:1.322382  
##  Median :  1.000   Median :   9.95   Median :2.956879  
##  Mean   :  1.038   Mean   :  14.00   Mean   :3.086623  
##  3rd Qu.:  1.000   3rd Qu.:  15.24   3rd Qu.:4.711841  
##  Max.   :100.000   Max.   :5010.66   Max.   :7.058179
```

```
head(orders_data)
```

```
##    id   orddate ordnum category qty     price        t
## 1 957 10FEB2008  38650       35   1  5.010658 6.789870
## 2 957 15MAR2008  48972       40   1 25.539017 6.696783
## 3 957 22NOV2008 150011       40   1 14.316170 6.006845
## 4 957 03OCT2009 286151       19   1 15.313187 5.144422
## 5 957 04APR2010 376779       14   1 12.782295 4.643395
## 6 957 14AUG2011 622093       99   1  8.691956 3.282683
```

```
RFM_table <- orders_data %>%
  group_by(id) %>%
  summarise(
    tof = max(t),
    r = min(t),
    f = n_distinct(ordnum),
    m = sum(price * qty)
    )

summary(RFM_table)
```

```
##        id                tof                  r                   f          
##  Min.   :     957   Min.   :0.002738   Min.   :0.002738   Min.   :  1.000  
##  1st Qu.: 4448960   1st Qu.:1.338809   1st Qu.:0.303901   1st Qu.:  2.000  
##  Median : 8090750   Median :3.800137   Median :0.851472   Median :  4.000  
##  Mean   : 8563488   Mean   :3.681231   Mean   :1.439085   Mean   :  6.111  
##  3rd Qu.:13378724   3rd Qu.:6.036961   3rd Qu.:2.031485   3rd Qu.:  8.000  
##  Max.   :16456238   Max.   :7.058179   Max.   :7.058179   Max.   :160.000  
##        m           
##  Min.   :    0.00  
##  1st Qu.:   18.95  
##  Median :   45.80  
##  Mean   :   88.64  
##  3rd Qu.:  103.75  
##  Max.   :26564.51
```

```
merged_data <- customer %>%
inner_join(RFM_table, by = "id")
train_data <- merged_data %>% filter(train == 1)
model <- lm(logtarg ~ log(tof) + log(r) + log(f) + log(m + 1), data = train_data)
summary(model)
```

```
##
## Call:
## lm(formula = logtarg ~ log(tof) + log(r) + log(f) + log(m + 1),
##     data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9643 -0.3745 -0.2178 -0.0539  5.5507
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.12108    0.05385   2.249  0.02458 *
## log(tof)    -0.06006    0.02063  -2.912  0.00361 **
## log(r)      -0.07702    0.01298  -5.935 3.12e-09 ***
## log(f)       0.18231    0.02787   6.541 6.65e-11 ***
## log(m + 1)  -0.01707    0.02010  -0.849  0.39574
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9151 on 5546 degrees of freedom
## Multiple R-squared:  0.05224,    Adjusted R-squared:  0.05156
## F-statistic: 76.42 on 4 and 5546 DF,  p-value: < 2.2e-16
```

```
test_data <- merged_data %>% filter(train == 0)
predicted_logtarg <- predict(model, newdata = test_data)
mse <- mean((test_data$logtarg - predicted_logtarg)^2)
print(paste("Mean Squared Error on the test set:", mse))
```

```
## [1] "Mean Squared Error on the test set: 0.80997002836259"
```

```
gains = function(yhat, respond, amt, ngrp=5){
  ans = data.frame(amt=amt, respond=respond, qtile=
  cut(yhat, breaks=quantile(yhat, probs=seq(0,1, 1/ngrp)),
  labels=paste("Q",ngrp:1, sep=""), include.lowest = T)
  ) %>%
  group_by(qtile) %>%
  summarise(n=n(), Nrespond=sum(respond), amt=sum(amt),
    RespRate=Nrespond/n, AvgAmt=amt/n) %>%
  arrange(desc(qtile)) %>%
  mutate(CumN=cumsum(n), CumResp=cumsum(Nrespond), CumAmt=cumsum(amt),
    CumRespRate=CumResp/CumN, CumAvgAmt=CumAmt/CumN)
  ans %>% mutate(liftResp=CumRespRate/CumRespRate[nrow(ans)],
    liftAmt=CumAvgAmt/CumAvgAmt[nrow(ans)])
}
```

```
summary(predicted_logtarg)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.22066  0.09609  0.25033  0.25750  0.40679  1.23016
```

```
train_data$buy <- ifelse(train_data$target>0, 1, 0)
test_data$buy <- ifelse(test_data$target>0, 1, 0)
gains_model1 <- gains(predicted_logtarg, test_data$buy, test_data$target)
gains_model1
```

```
## # A tibble: 5 x 13
##   qtile     n Nrespond    amt RespRate AvgAmt  CumN CumResp CumAmt CumRespRate
##   <fct> <int>    <dbl>  <dbl>    <dbl>  <dbl> <int>   <dbl>  <dbl>       <dbl>
## 1 Q1     2246      405 18384.   0.180   8.19   2246     405 18384.      0.180
## 2 Q2     2246      178  7534.   0.0793  3.35   4492     583 25919.      0.130
## 3 Q3     2246      107  5021.   0.0476  2.24   6738     690 30940.      0.102
## 4 Q4     2246       63  3369.   0.0280  1.50   8984     753 34309.      0.0838
## 5 Q5     2246       52  2243.   0.0232  0.999 11230     805 36552.      0.0717
## # i 3 more variables: CumAvgAmt <dbl>, liftResp <dbl>, liftAmt <dbl>
```

The money expect to make per customer to select 40% of the names to be contacted is

```r
25918.7/(2246*2)
```

```
## [1] 5.769969
```

We see that the Cumulative respond for this gains table at 40% is 583, so the response rate is

```r
583/4492
```

```
## [1] 0.1297863
```

```r
#or the average respond rate of the 20th and 40th quantile
(0.18032+0.079252)/2
```

```
## [1] 0.129786
```

We have a highly unbalanced case here, considering adding weights

```r
sum(train_data$buy)/nrow(train_data)
```

```
## [1] 0.07422086
```

```r
print(paste("The proportion of buy class equals to 1", sum(train_data$buy)/nrow(train_data)))
```

```
## [1] "The proportion of buy class equals to 1 0.074220861106107"
```

```r
weights <-  ifelse(train_data$buy == 1,
                  1/(mean(train_data$buy == 1)),
                  1/(mean(train_data$buy == 0)))
lm2 <- glm(buy ~ log(tof)+log(r)+log(f)+log(m+1), data = train_data, weights = weights)
```

Accuracy matrix for lm2 (weights set)

```r
pred_prob <- predict(lm2, newdata = test_data[, c('tof', 'r', 'f', 'm')], type = "response")
pred_class <- ifelse(pred_prob > 0.5, 1, 0)
conf_matrix <- table(Actual = test_data$buy, Predicted = pred_class)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
sensitivity <- conf_matrix[2,2] / sum(conf_matrix[2,])
specificity <- conf_matrix[1,1] / sum(conf_matrix[1,])
conf_matrix
```

```
##       Predicted
## Actual    0    1
##      0 6999 3426
##      1  251  554
```

```r
print("Model Evaluation Metrics:")
```

```
## [1] "Model Evaluation Metrics:"
```

```r
print(paste("Accuracy:", round(accuracy, 3)))
```

```
## [1] "Accuracy: 0.673"
```

```
print(paste("Conversion Probability:", round(sensitivity, 3)))
```

```
## [1] "Conversion Probability: 0.688"
```

```
print(paste("True-negative:", round(specificity, 3)))
```

```
## [1] "True-negative: 0.671"
```

```
summary(lm2)
```

```
##
## Call:
## glm(formula = buy ~ log(tof) + log(r) + log(f) + log(m + 1),
##     data = train_data, weights = weights)
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.369487   0.028819  12.821  < 2e-16 ***
## log(tof)    -0.083486   0.009725  -8.585  < 2e-16 ***
## log(r)      -0.057141   0.005923  -9.647  < 2e-16 ***
## log(f)       0.197419   0.014225  13.879  < 2e-16 ***
## log(m + 1)  -0.035514   0.010800  -3.288  0.00101 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.4160092)
##
##     Null deviance: 2775.5  on 5550  degrees of freedom
## Residual deviance: 2307.2  on 5546  degrees of freedom
## AIC: 9423.8
##
## Number of Fisher Scoring iterations: 2
```

We see from the output, log(tof), log(r), log(f) all have p-value less than 2e-16. The variable with the smallest t value is log(f), so the log of frequency of orders is the most predictive variable.

```
lm3 <- glm(logtarg~log(tof)+log(r)+log(f)+log(m+1), data = train_data, subset = buy==1)
summary(lm3)
```

```
##
## Call:
## glm(formula = logtarg ~ log(tof) + log(r) + log(f) + log(m +
##     1), data = train_data, subset = buy == 1)
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.861944   0.192956  14.832   <2e-16 ***
## log(tof)    -0.047932   0.057156  -0.839   0.4022
## log(r)       0.007575   0.033440   0.227   0.8209
## log(f)      -0.140041   0.093456  -1.498   0.1348
## log(m + 1)   0.219094   0.072862   3.007   0.0028 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.5643824)
```

```
##
##     Null deviance: 236.26  on 411  degrees of freedom
## Residual deviance: 229.70  on 407  degrees of freedom
## AIC: 940.5
##
## Number of Fisher Scoring iterations: 2
```

```r
y_hat <- predict(lm3, newdata = test_data[, c('tof', 'r', 'f', 'm')])
```

```r
conversion_yhat = pred_prob*y_hat
gains_model2 <- gains(conversion_yhat, test_data$buy, test_data$target)
gains_model2
```

```
## # A tibble: 5 x 13
##   qtile     n Nrespond    amt RespRate AvgAmt  CumN CumResp CumAmt CumRespRate
##   <fct> <int>    <dbl>  <dbl>    <dbl>  <dbl> <int>   <dbl>  <dbl>       <dbl>
## 1 Q1     2246      403 18641.   0.179   8.30   2246     403 18641.      0.179
## 2 Q2     2246      181  7130.   0.0806  3.17   4492     584 25771.      0.130
## 3 Q3     2246      111  5580.   0.0494  2.48   6738     695 31351.      0.103
## 4 Q4     2246       58  3128.   0.0258  1.39   8984     753 34479.      0.0838
## 5 Q5     2246       52  2073.   0.0232  0.923 11230     805 36552.      0.0717
## # i 3 more variables: CumAvgAmt <dbl>, liftResp <dbl>, liftAmt <dbl>
```

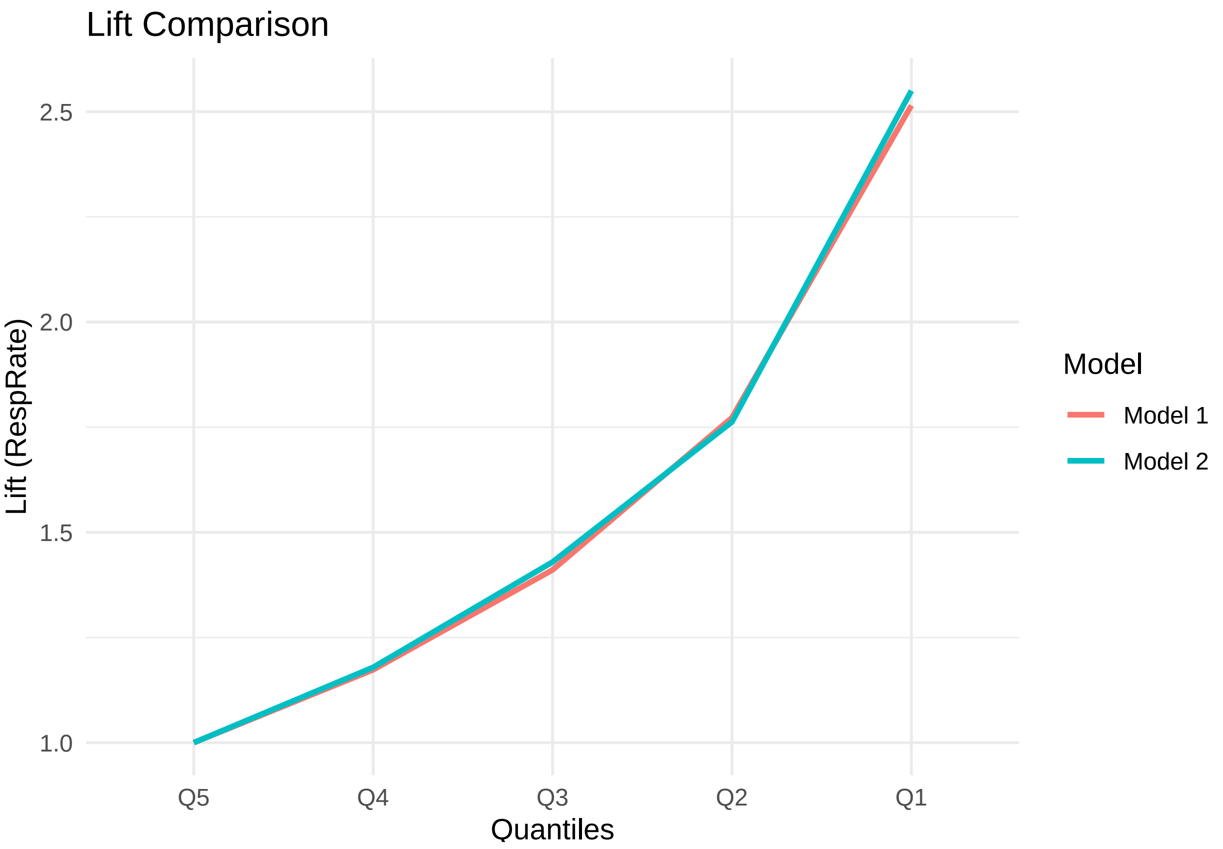
```r
gains_model1
```

```
## # A tibble: 5 x 13
##   qtile     n Nrespond    amt RespRate AvgAmt  CumN CumResp CumAmt CumRespRate
##   <fct> <int>    <dbl>  <dbl>    <dbl>  <dbl> <int>   <dbl>  <dbl>       <dbl>
## 1 Q1     2246      405 18384.   0.180   8.19   2246     405 18384.      0.180
## 2 Q2     2246      178  7534.   0.0793  3.35   4492     583 25919.      0.130
## 3 Q3     2246      107  5021.   0.0476  2.24   6738     690 30940.      0.102
## 4 Q4     2246       63  3369.   0.0280  1.50   8984     753 34309.      0.0838
## 5 Q5     2246       52  2243.   0.0232  0.999 11230     805 36552.      0.0717
## # i 3 more variables: CumAvgAmt <dbl>, liftResp <dbl>, liftAmt <dbl>
```

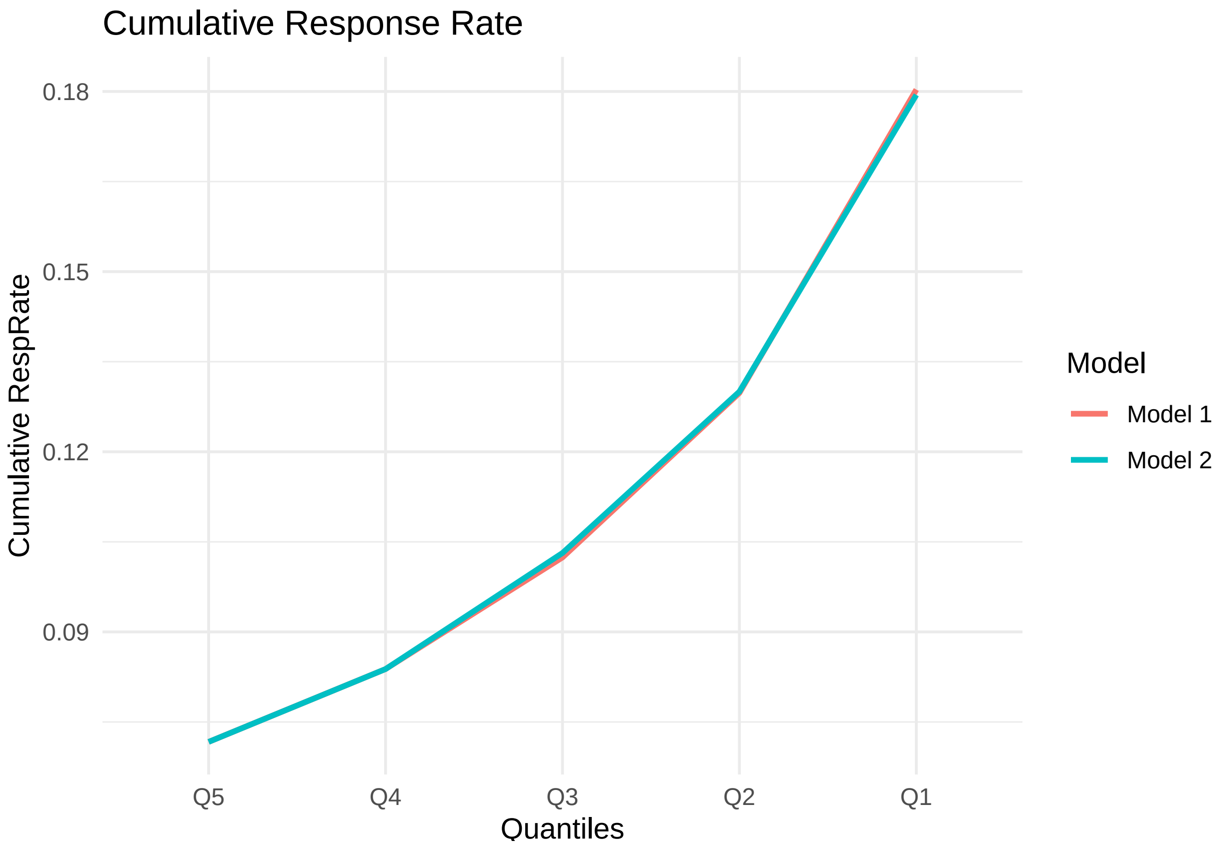
At the 40% threshold (cumulative population with `CumN = 6738`), Model 2 achieves a cumulative response rate (`CumRespRate`) of 0.1031 and a lift (`liftResp`) of 1.44, while Model 1 achieves a slightly lower cumulative response rate of 0.1024 and lift of 1.43. These results indicate that Model 2 marginally outperforms Model 1 in prioritizing responses within the top 40% of the population. Based on these metrics, Model 2 demonstrates better performance for optimizing responses at this threshold, making it the preferable choice for scenarios focused on maximizing responses in this segment.

```r
# Plot Lift Comparison
library(ggplot2)

gains_model1$Model <- "Model 1"
gains_model2$Model <- "Model 2"

combined_gains <- rbind(gains_model1, gains_model2)

# Plot Lift
ggplot(combined_gains, aes(x = qtile, y = liftAmt, color = Model, group = Model)) +
  geom_line(size = 1) +
  labs(title = "Lift Comparison", x = "Quantiles", y = "Lift (RespRate)") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```
# Cumulative Response Rate Comparison
ggplot(combined_gains, aes(x = qtile, y = CumRespRate, color = Model, group = Model)) +
  geom_line(size = 1) +
  labs(title = "Cumulative Response Rate", x = "Quantiles", y = "Cumulative RespRate") +
  theme_minimal()
```


Cumulative Response Rate
Cumulative RespRate
0.18
0.15
0.12
0.09
Q5
Q4
Q3
Q2
Q1
Quantiles
Model
Model 1
Model 2