Class 11: Upflift Modeling in R

Load packages and read in data

```r
### Load packages:
library(knitr)
library(janitor)
library(mktg482)
library(tidyverse)
library(splitstackshape)

### Read in data:
set.seed(3456)
rm(list=(ls()))
load("../../Data/uplift_example.Rdata")
str(campaign)
```

```
'data.frame':   1000 obs. of  22 variables:
 $ y    : num  0 0 0 0 0 0 1 1 0 0 ...
 $ treat: num  0 0 0 1 0 0 0 0 0 0 ...
 $ X1   : num  -0.8391 0.0677 -2.9104 2.1739 0.2013 ...
 $ X2   : num  -1.184 0.197 1.662 -0.47 0.583 ...
 $ X3   : num  -0.619 -1.905 0.632 -0.673 0.112 ...
 $ X4   : num  0.92 -1.14 -0.834 -0.711 0.541 ...
 $ X5   : num  0.6467 0.7788 0.6021 -0.125 -0.0968 ...
 $ X6   : num  -0.864 0.583 2.25 0.259 -0.303 ...
 $ X7   : num  0.682 0.117 -0.534 -1.762 -0.24 ...
 $ X8   : num  0.941 1.294 -0.37 1.176 0.5 ...
 $ X9   : num  -0.0747 -0.5936 -0.1688 0.4976 1.3687 ...
 $ X10  : num  0.3 0.527 -1.708 1.237 0.33 ...
 $ X11  : num  1.756 -0.191 -0.434 1.769 0.371 ...
 $ X12  : num  -1.461 -0.28 -0.328 -0.468 -0.966 ...
 $ X13  : num  0.2215 -0.6734 1.0292 0.359 -0.0711 ...
 $ X14  : num  -1.95 0.178 -0.683 1.168 -0.756 ...
 $ X15  : num  1.95 -1.266 1.98 -0.161 -1.677 ...
 $ X16  : num  0.364 -1.112 0.183 -1.595 0.125 ...
 $ X17  : num  -1.897 1.068 1.375 -0.399 2.292 ...
 $ X18  : num  -0.2 -1.26 0.354 -0.285 1.415 ...
 $ X19  : num  0.17 -0.155 0.772 0.616 2.043 ...
 $ X20  : num  -0.8033 -0.0421 -0.8777 -0.7406 -0.3874 ...
```

Please note that this data is stacked, that is, treatment and control data are in the same dataframe with a binary `treat` variable that indicates whether the observation is in the treatment or control group. This will be important when we work on Part II of the Creative Gaming case because we will need to stack the data as well.

Create a train/test split in the uplift data

This is a little tricky because we need to randomize while keeping the proportions of all four possibilities (i.e., whether a customer is in treatment or control and whether the customer bought or did not by as indicated by the `treat` and `y` variables respectively) the same in the training data and the test data as it was in the overall data. We use the `stratified()` function from the `splitstackshape` package to achieve this specifying that we want a 70/30 split using the `size` argument:

```
split.index <- stratified(campaign, group=c("treat", "y"), size=0.7, bothSets=TRUE)
campaign.train <- split.index[[1]]
campaign.test <- split.index[[2]]
```

We can check that randomization kept the proportions pretty even:

```
campaign %>% tabyl(y, treat) %>% adorn_percentages("all")
```

```
 y     0     1
 0 0.261 0.229
 1 0.239 0.271
```

```
campaign.train %>% tabyl(y, treat) %>% adorn_percentages("all")
```

```
 y         0         1
 0 0.2614286 0.2285714
 1 0.2385714 0.2714286
```

```
campaign.test %>% tabyl(y, treat) %>% adorn_percentages("all")
```

```
 y    0    1
 0 0.26 0.23
 1 0.24 0.27
```

Run seperate logistic regressions for the treatment and control sample

Note: we exclude the treatment indicator (it is constant for each sample and it is not something we want to use to predict purchase).

```
fm <- formula(y ~ . - treat)
lr_treatment <- glm(fm, data=campaign.train %>% filter(treat==1), family=binomial)
lr_control <- glm(fm, data=campaign.train %>% filter(treat==0), family=binomial)
```

Create predictions and the uplift score for each customer in the test data

For each customer in the test data (regardless of whether they were in the treatment group or the control group), we get `Prob(purchase | treated)` and `Prob(purchase | not treated)` from the above models and then take their difference (i.e., the uplift score):

```
campaign.test <- campaign.test %>%
  mutate(pred_treat = predict(lr_treatment, newdata=campaign.test, type="response"),
         pred_control = predict(lr_control, newdata=campaign.test, type="response"),
         uplift_score = pred_treat - pred_control)
```

Let's look at the data

We sort the data on uplift score from highest to lowest and select a few key columns.

```
campaign.test %>%
  arrange(-uplift_score) %>%
  select(y, treat, pred_treat, pred_control, uplift_score) %>%
  head()
```

```
  y treat pred_treat pred_control uplift_score
1 1     1  0.9951174   0.01561716    0.9795003
2 0     0  0.9796483   0.01088566    0.9687627
3 1     1  0.9717307   0.01802790    0.9537028
4 0     0  0.9536232   0.01405489    0.9395683
5 1     1  0.9971628   0.05782007    0.9393427
6 0     0  0.9222191   0.02786441    0.8943547
```

Evaluate the performance of the uplift model

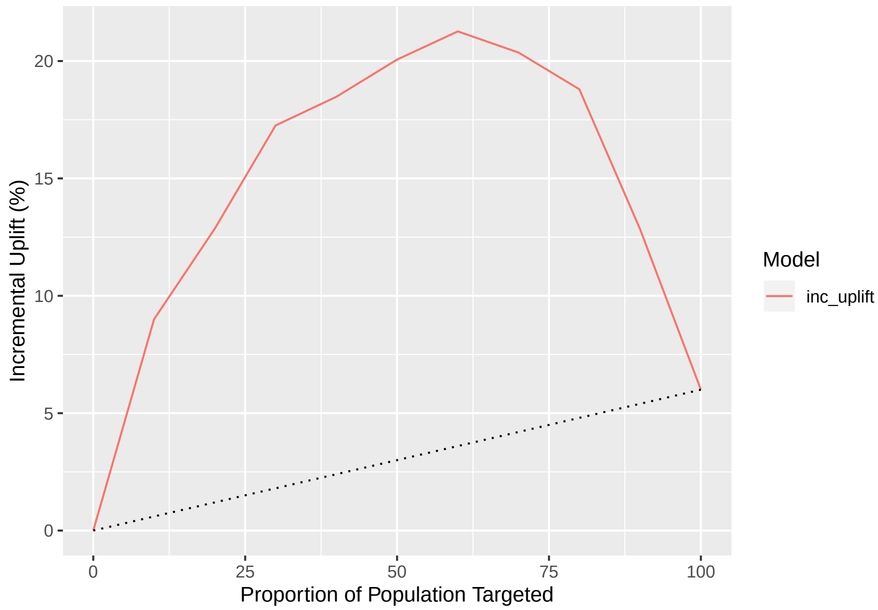
To calcuate the performance measures we discussed in class, we can use the `QiniTable()` function in the `mktg482` package (note: this is an improved version of a function by the same name in the `tools4uplift` package which you do not need):

```
PerfTable_uplift <- QiniTable(
                                campaign.test,
                                treat = "treat",
                                outcome = "y",
                                prediction = "uplift_score",
                                nb.group = 10
                              )
PerfTable_uplift
```

```
   cum_per T_Y1 T_n C_Y1 C_n incremental_Y1 inc_uplift      uplift
1      0.1   15  15    1  10       13.50000    9.00000  0.90000000
2      0.2   28  30    9  31       19.29032   12.86022  0.48571429
3      0.3   40  45   16  51       25.88235   17.25490  0.45000000
4      0.4   50  60   26  70       27.71429   18.47619  0.14035088
5      0.5   58  75   32  86       30.09302   20.06202  0.15833333
6      0.6   66  90   36  95       31.89474   21.26316  0.08888889
7      0.7   71 105   42 109       30.54128   20.36086 -0.09523810
8      0.8   76 120   49 123       28.19512   18.79675 -0.16666667
9      0.9   80 135   63 140       19.25000   12.83333 -0.55686275
10     1.0   81 150   72 150        9.00000    6.00000 -0.83333333
```

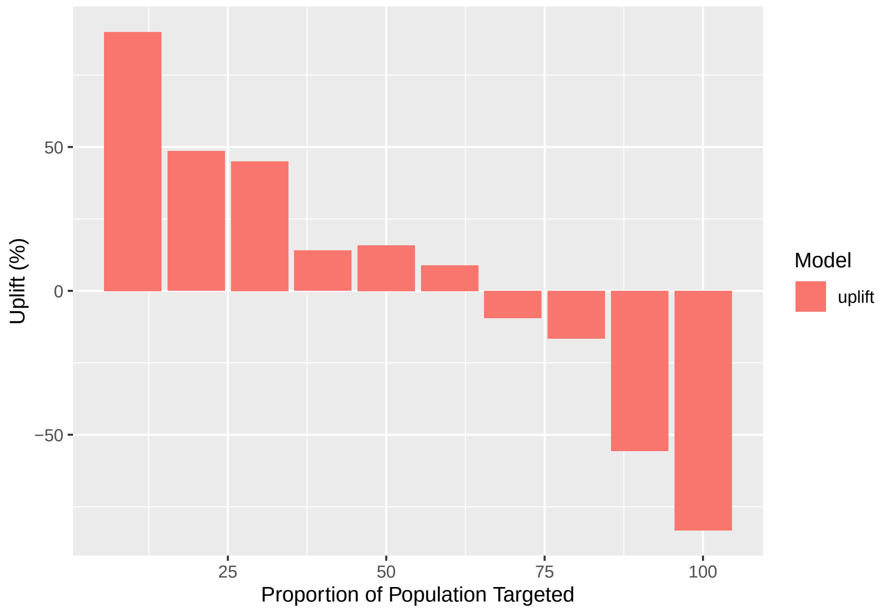
We can use the the `QiniCurve()` function to plot Incremental Uplift (%) (i.e., Incremental_Y1 / Total Number Treated):

```
QiniCurve(PerfTable_uplift)
```



Finally, we can use the `QiniBarPlot()` function to plot Uplift (%) (i.e., the difference between Treatment and Control conversion percentages):

```
QiniBarPlot(PerfTable_uplift)
```



Compare performance of the uplift model to a propensity model

We not compare the performance of our uplift model to a plain vanilla propensity model. We can easily obtain the performance table for the propensity model by specifying `prediction = "pred_treat"`

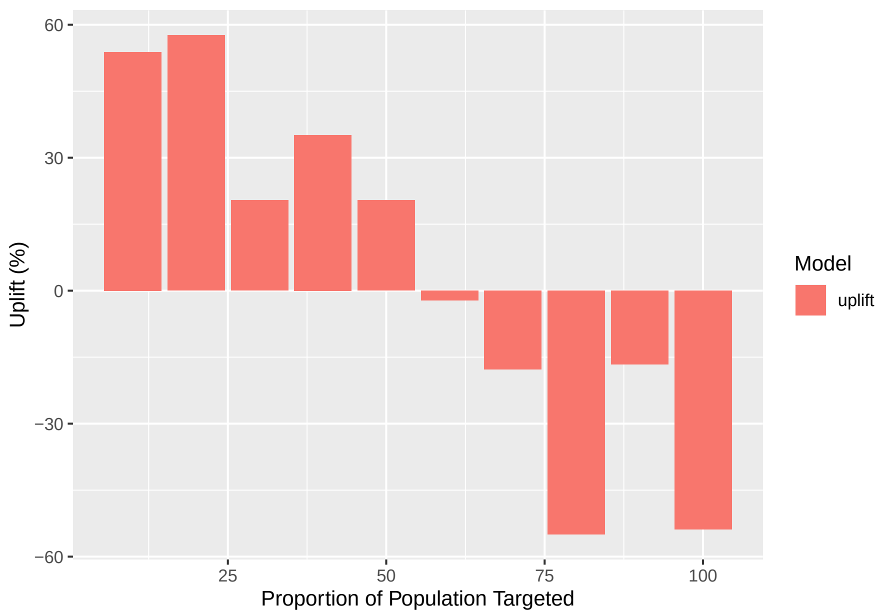
```
PerfTable_propensity <- QiniTable(
                                  campaign.test,
                                  treat = "treat",
                                  outcome = "y",
                                  prediction = "pred_treat",
```

```r
                                    nb.group = 10
                                  )
PerfTable_propensity
```

```
   cum_per T_Y1 T_n C_Y1 C_n incremental_Y1 inc_uplift      uplift
1      0.1   15  15    6  13       8.076923   5.384615  0.53846154
2      0.2   29  30   11  27      16.777778  11.185185  0.57619048
3      0.3   40  45   20  44      19.545455  13.030303  0.20392157
4      0.4   50  60   26  63      25.238095  16.825397  0.35087719
5      0.5   61  75   35  80      28.187500  18.791667  0.20392157
6      0.6   69  90   40  89      28.550562  19.033708 -0.02222222
7      0.7   73 105   48 107      25.897196  17.264798 -0.17777778
8      0.8   76 120   54 115      19.652174  13.101449 -0.55000000
9      0.9   81 135   65 137      16.948905  11.299270 -0.16666667
10     1.0   81 150   72 150       9.000000   6.000000 -0.53846154
```

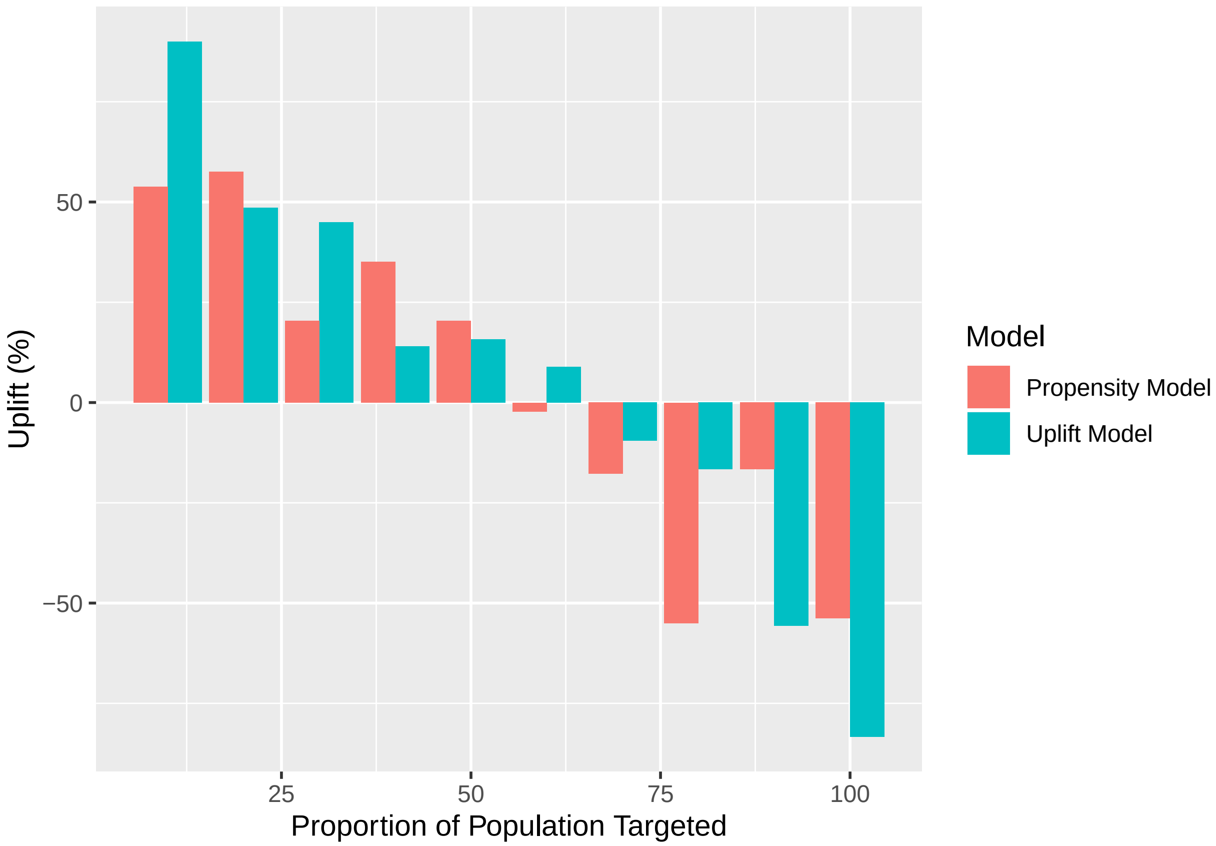
Let's plot plot Uplift (%):

```r
QiniBarPlot(PerfTable_propensity)
```



It is helpful to compare the multiple models on the same graph. The `QiniBarPlot()` function can do this:

```r
QiniBarPlot(PerfTable_uplift,
            PerfTable_propensity,
            modelnames = c("Uplift Model", "Propensity Model"))
```



Notice that the uplift model is better places customers with high incrementality in earlier deciles. The propensity model does not do terribly; this is because in this data, the customers who have the best propensity also happen to have the best uplift:

```
cor(campaign.test$pred_treat, campaign.test$uplift_score)
```

```
[1] 0.8484914
```