Targeting Based on Heterogeneous Treatment Effects

Drew Ficken

June 4th, 2019

Contents

1 Important: Read this first **2**

2 Managing a computationally intensive data-analysis **3**

3 Overview **4**

4 Summary of findings **5**

5 Step 1: Estimation and prediction of conditional average treatment effects **6**
5.1 Data pre-processing . . . 6
5.2 Randomization checks . . . 6
5.3 Estimation of heterogeneous treatment effects . . . 7
5.4 Estimation: Hints . . . 8
5.5 Predict treatment effects . . . 9

6 Step 2: Model fit and profit evaluation in 2017 validation sample **11**
6.1 Descriptive analysis of predicted treatment effects . . . 11
6.2 Model validation: Lifts . . . 13
6.3 Profit predictions . . . 16
6.4 Profits from targeting the top n percent of customers . . . 16

7 Step 3: How well do the models predict out-of-sample? — Profits and external model validity **20**

8 Optional analysis (bonus) **29**

```
setwd("~/Google Drive/Booth/Data Science for Marketing/Week 10")
```

1 Important: Read this first

- Please recall that this is an **individual assignment**, **not a group assignment**.
- Discussing the assignment with other students before the due date is a violation of the honor code. Also, please do not reply to questions on Canvas concerning assignment 7.
- Please submit the assignment by **midnight on Wednesday, June 12, 2019**.
- Please do not start to work on the assignment one day before the due date. Give it some time, and work on the different parts systematically, step by step. Breaking the overall task into small pieces and writing clean code will make the analysis much easier.

2 Managing a computationally intensive data-analysis

Some of the analysis will take a substantial amount of time. To be efficient, save the results from the computationally intensive steps, such as the model `fit` objects or the predicted outputs, to a file.

Also, when you conduct a long, extensive analysis, it is useful to spread the code across multiple scripts or R Markdown files. For example, you may want to create three R Markdown files corresponding to the three steps in the analysis outlined below.

Here is a useful trick to automatically save the results when you knit an R Markdown file. Use the `cache` option in the header of a code chunk:

```
{r, cache = TRUE}
```

R will then save all computed objects in that code chunk (not *all* code chunks unless you set `cache = TRUE` for all chunks) to disk and automatically load the objects if you knit the R Markdown file again. If you modify the code chunk, or if you delete the files that are created to store the objects, R will re-generate the results when you knit the R Markdown file.

3 Overview

As in the previous assignment, we use customer-level data from the data base of a company that sells kitchenware, housewares, and specialty food products. We again use the October 2017 sample that contains records on 250,000 customers, and in addition we use a sample from a similar catalog mailing in October 2018 that contains 125,000 records.

Both data sets have a very important property: The catalog-mailing in the October 2017 and 2018 data was fully **randomized**—the treatment, `mailing_indicator`, was randomly assigned to the customers.

The randomized treatment assignment and the rich customer-level data are crucial to achieve our goals:

1. Predict **incremental dollar spending** that we can *attribute* to the catalog mailing. Formally, we predict
$$\tau_i = \mathbb{E}[Y_i(1) - Y_i(0)|\boldsymbol{x}_i],$$
the conditional average treatment effect of a catalog mailing. This heterogeneous treatment effect, τ_i, is the expected change in spending due to the catalog mailing. In particular, we predict incremental dollar spending at the **customer-level**, based on all the features (variables) that capture past customer behavior.
2. Predict customer-level incremental profits due to targeting a customer (catalog mailing), and develop a targeting strategy.
3. Evaluate the predictive power and profitability of the targeting strategy. Compare different targeting strategies, based on different estimation methods or heuristics.
4. Evaluate the predictive power and profitability of the targeting strategy that was developed using the 2017 data in an implementation of the strategy in 2018. This evaluation allows us to assess the strict external validity or transportability of the results. Using more straightforward language, it allows us to assess if the fancy techniques that we are using actually work.

The process in steps 1-4 solves the two biggest issues in CRM (or marketing analytics in general): Create a **fully personalized** targeting strategy with customer segments of one, and evaluate the profitability based on the **incremental effect** of the targeting strategy, not based on the *level* of sales and profits that would have been achieved even without any targeting efforts.

Detailed hints on how to conduct the analysis are provided below.

4 Summary of findings

In your write-up, first **summarize your key findings** and the main insights that you gained from the analysis. Please be concise and provide a summary that does not exceed one page.

```
library(bit64)
library(data.table)
library(glmnet)
library(ggplot2)
library(knitr)
```

5 Step 1: Estimation and prediction of conditional average treatment effects

We use the 2017 data to estimate and validate several models to predict the heterogeneous treatment effects.

```
# load('/classes/3710501_spr2019/Data/Assignment-7/Customer-Development-2017.RData')
load("Customer-Development-2017.RData")
```

Split the sample into a 50 percent training and 50 percent validation sample. Please make sure to **use the same seed** as in the code chunk below so that we have roughly comparable results.

```
set.seed(2001)
crm_DT[, `:=`(training_sample, rbinom(nrow(crm_DT), 1, 0.5))]
```

To make the code more readable, I recommend to rename the `mailing_indicator` to make it clear that the randomized mailing is the treatment, W_i.

```
setnames(crm_DT, "mailing_indicator", "W")
```

5.1 Data pre-processing

As in the previous assignment, remove highly correlated features from the data set.

```
### grab code from assignment 6
cor_matrix = cor(crm_DT[, !c("customer_id", "W", "outcome_spend"),
    with = FALSE])
cor_matrix[upper.tri(cor_matrix, diag = TRUE)] = NA

cor_DT = data.table(row = rep(rownames(cor_matrix), ncol(cor_matrix)),
    col = rep(colnames(cor_matrix), each = ncol(cor_matrix)),
    cor = as.vector(cor_matrix))
cor_DT = cor_DT[is.na(cor) == FALSE]

large_cor_DT = cor_DT[abs(cor) > 0.95]

crm_DT = crm_DT[, !large_cor_DT$row, with = FALSE]
```

5.2 Randomization checks

Inspect the data to estimate the probability of a mailing,

$$e = \Pr\{W_i = 1\}.$$

Recall that e is called the *propensity score*.

Perform a quick check to assess if the treatment (catalog mailing) was indeed randomized in the whole data base and hence does not depend on the customer attributes, $\boldsymbol{x}_i$.

```
# Find probability of mailing

table(crm_DT$W)
```

```
     0      1
 82784 167216
```

```
# Check for randomization
check_randomized = lm(W ~ . - customer_id - outcome_spend - training_sample,
    data = crm_DT)
# summary(check_randomized)
```

The probability of a customer receiving a mailing within the dataset is approximately 2/3, with a total of 167,216 customers receiving the "treatment", and about half of that number (82,784) not receiving the mailing or "treatment".

In looking at the regression output on "W" we can see that there are very few significant covariates, and the ones that do register as significant most likely fall under the "false positive" narrative. This means there is very little correlation between mailings and the other independent variables. Thus, it's pretty clear that the treatment was indeed randomized in the whole data base and does not depend on the customer attributes.

5.3 Estimation of heterogeneous treatment effects

We use the training sample to estimate the conditional average treatment effect on dollar spending, τ_i, due to catalog targeting.

Estimate **linear models with treatment-interactions**:

(a) OLS
(b) LASSO

We also use a recently developed non-parametric estimator that directly predicts the CATE for each customer:

(c) Causal forest

Estimating a causal forest is computationally much more intensive than, for example, estimating a LASSO. To avoid crashes because you run out of memory on your computer or because the `rstudio-class` server is heavily used, the *predicted* conditional average treatment effects from a causal forest are included in the file `Predicted-Causal-Forest-CATE.RData`. The causal forest was estimated using 1000 trees.

The file includes two data tables with predictions for the 2017 and 2018 samples, `predict_DT_2017` and `predict_DT_2018`. The conditional average treatment effect predictions are in the column `tau_cforest`. The tables also contain the `customer_id` for each observation, which you can use to merge the prediction data.

Recall that you can include **interactions between variables in an R model formula** using either the syntax `x*z` or `x:z`. When you use `x*z`, R will include `x`, `z`, and the interaction (`x` multiplied with `z`) in the regression. Using `x:z`, only the interaction is included. Similarly, using `.*z`, all variables in the data set and all interactions of the variables with `z` will be included, while `.:z` only adds the interaction terms with `z`.

5.4 Estimation: Hints

To simplify your code I recommend to **create separate training and validation samples** from the full data set. Exclude the variables that are not used in the statistical analysis. As a result, there will be no need to explicitly exclude some variables when estimating the different models.

```
training_DT = crm_DT[training_sample == 1, !c("customer_id",
    "training_sample"), with = FALSE]
validation_DT = crm_DT[training_sample == 0, !c("customer_id",
    "training_sample"), with = FALSE]
```

For replicability, you may **provide glmnet with the folds used for cross-validation**. Set a seed and then draw numbers indicating the fold $(1, 2, \ldots, 10)$ that an observation belongs to:

```
set.seed(961)

N_obs_training = nrow(training_DT)
folds = sample(1:10, N_obs_training, replace = TRUE)
```

You can now use these fold id's in `glmnet` as follows:

```
fit_glmnet = cv.glmnet(x = X, y = y, nfolds = 10, foldid = folds)
```

Once you have estimated the models, save all the output objects in a file for later use.

```
########### Linear Models with treatment interactions ###############

### OLS

# fit_OLS = lm(outcome_spend ~ .*W, data = training_DT)
# saveRDS(fit_OLS, 'fit_OLS')

fit_OLS = readRDS("fit_OLS")
summary_OLS = summary(fit_OLS)

results = data.table(input = rownames(summary_OLS$coefficients),
    est_OLS = summary_OLS$coefficients[, 1], p_OLS = summary_OLS$coefficients[,
        4])



### LASSO

# make the model matrix for the LASSO

X = model.matrix(outcome_spend ~ 0 + . * W, data = training_DT)

y = training_DT$outcome_spend

# fit_LASSO = cv.glmnet(x=X, y=y, nfolds = 10, foldid =
# folds) # alpha = 1 is standard saveRDS(fit_LASSO,
# 'fit_LASSO')

fit_LASSO = readRDS("fit_LASSO")
```

```
results[, `:=`(est_LASSO, coef(fit_LASSO, s = "lambda.min")[,
    1])]

############ Non-parametric Estimator ####################

### Causal Forest

# the *predicted* conditional average treatment effects from
# a causal forest are included in the file
# `Predicted-Causal-Forest-CATE.RData`.  The causal forest
# was estimated using 1000 trees.

load("Predicted-Causal-Forest-CATE.RData")
```

5.5 Predict treatment effects

To validate the models you need to predict the treatment effects. You will make your life much easier if you first create a new data.table,

```
predict_DT = crm_DT[training_sample == 0, c("customer_id", "outcome_spend",
    "W"), with = FALSE]

### add predictions for OLS
predict_DT[, `:=`(y_OLS, predict(fit_OLS, newdata = validation_DT))]

### add predictions for LASSO
new_X = model.matrix(outcome_spend ~ 0 + . * W, data = validation_DT)

predict_DT[, `:=`(y_LASSO, predict(fit_LASSO, newx = new_X, s = "lambda.min"))]

######## find individual heterogeneous treatment effects for the
######## following:

### OLS

# change dataset so that W = 1
W_valid = copy(validation_DT)
W_valid[, `:=`(W, 1)]
predict_W_OLS = predict(fit_OLS, newdata = W_valid)

# change dataset so that W = 0
no_W_valid = copy(validation_DT)
no_W_valid[, `:=`(W, 0)]
predict_no_W_OLS = predict(fit_OLS, newdata = no_W_valid)

# find heterogeneous treatment effect usng OLS
tau_OLS = predict_W_OLS - predict_no_W_OLS

### LASSO

# create model.matrix with W = 1
X_with_W = model.matrix(outcome_spend ~ 0 + . * W, data = W_valid)
predict_W_LASSO = predict(fit_LASSO, newx = X_with_W, s = "lambda.min")
```

```
# create model.matrix with W = 0
X_no_W = model.matrix(outcome_spend ~ 0 + . * W, data = no_W_valid)
predict_no_W_LASSO = predict(fit_LASSO, newx = X_no_W, s = "lambda.min")

# find heterogeneous treatment effect usng LASSO
tau_LASSO = predict_W_LASSO - predict_no_W_LASSO

predict_DT = cbind.data.frame(predict_DT, tau_OLS, tau_LASSO)

### merge Causal Forest with predict_DT
predict_DT = merge(predict_DT, predict_DT_2017)

### tau_LASSO is just vector not named vector so have to
### provide column name
colnames(predict_DT)[7] <- "tau_LASSO"

saveRDS(predict_DT, "predict_DT")
```

Add the model predictions to this table.

Also, merge the causal forest predictions using the `customer_id` as key.

Warning: When you merge the causal forest predictions using the common `customer_id` as key, `setkey` will change the order of the observations. Hence, it is necessary to perform the merge step last, *after you have added all predictions* to `predict_DT`. Otherwise, the order of `predict_DT` will not match the order of `validation_DT`, and the model predictions will not match the correct outcomes observed in the data.

Once all model predictions are added, save the table to a file for use in step 2.

6 Step 2: Model fit and profit evaluation in 2017 validation sample

6.1 Descriptive analysis of predicted treatment effects

Document the ATE (average treatment effect in the data). Summarize and graph the distribution of the predicted heterogeneous treatment effects, τ_i, from the different estimation methods. How much variation is there in the individual treatment effects compared to the ATE? Do some of the treatment effect distributions appear more "plausible" than others? How similar/dissimilar are the model predictions, and what is the degree of correlation across the predictions?

Also compare and comment on the scale difference between the treatment effects and the level of sales.

```
mean_spend_0 = mean(predict_DT[W == 0, outcome_spend])
mean_spend_1 = mean(predict_DT[W == 1, outcome_spend])
ATE = mean_spend_1 - mean_spend_0

cat(mean_spend_0, mean_spend_1, ATE)
```

```
5.803782 8.880369 3.076587
```

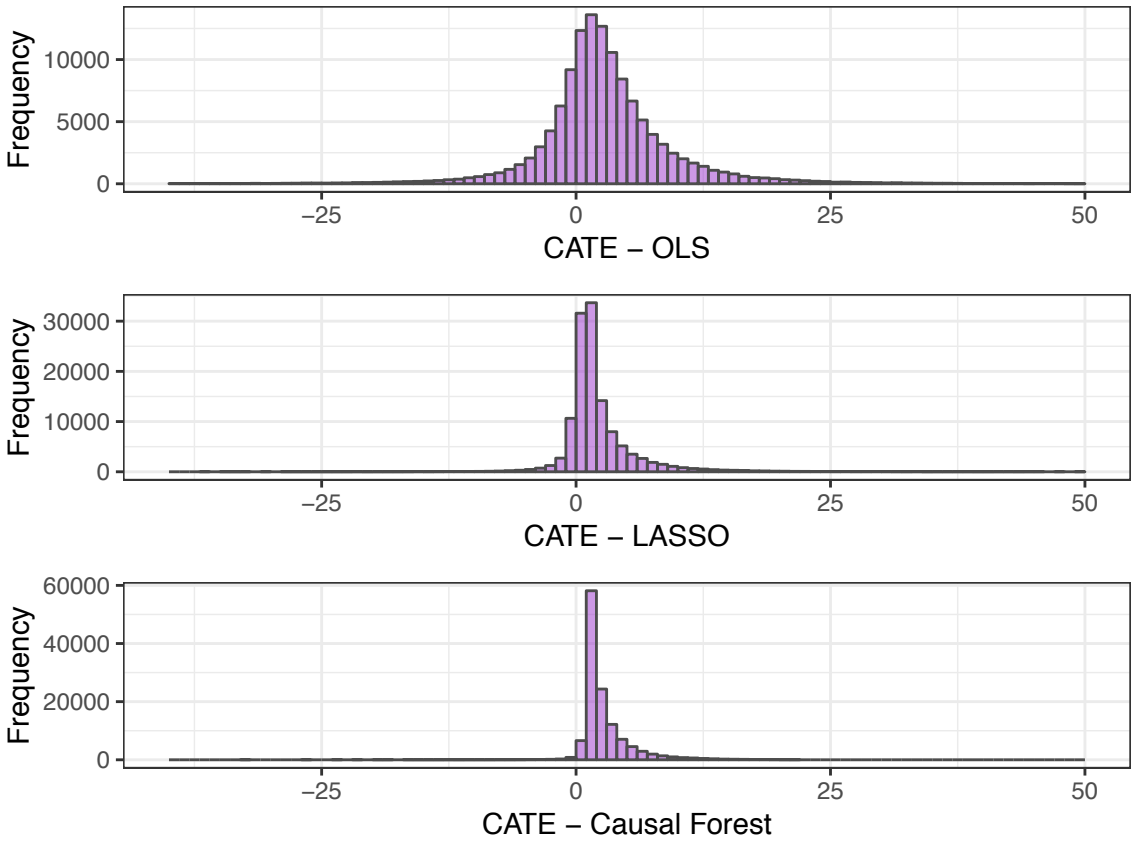
In looking at the Average Treatment Effect (ATE) in the 2017 validation sample, we see that on average the people that were sent a mailer ended up spending $3.08 more than the people that weren't. People that received the mailer spent on average $8.88, while the people that didn't receive the mailer only spent $5.80 on average.

```
# Summarize and graph the distribution of the predicted
# heterogeneous treatment effects, tau_i, from the different
# estimation methods.

require(gridExtra)

hist_OLS <- ggplot(predict_DT, aes(x = tau_OLS)) + geom_histogram(binwidth = 1,
    center = 0.5, color = "gray30", fill = "darkorchid3", alpha = 0.5) +
    scale_x_continuous("CATE - OLS", limits = c(-40, 50)) + scale_y_continuous("Frequency") +
    theme_bw()

hist_LASSO <- ggplot(predict_DT, aes(x = tau_LASSO)) + geom_histogram(binwidth = 1,
    center = 0.5, color = "gray30", fill = "darkorchid3", alpha = 0.5) +
    scale_x_continuous("CATE - LASSO", limits = c(-40, 50)) +
    scale_y_continuous("Frequency") + theme_bw()

hist_cforest <- ggplot(predict_DT, aes(x = tau_cforest)) + geom_histogram(binwidth = 1,
    center = 0.5, color = "gray30", fill = "darkorchid3", alpha = 0.5) +
    scale_x_continuous("CATE - Causal Forest", limits = c(-40,
        50)) + scale_y_continuous("Frequency") + theme_bw()

grid.arrange(hist_OLS, hist_LASSO, hist_cforest, ncol = 1, nrow = 3)
```

```
Warning: Removed 435 rows containing non-finite values (stat_bin).
```

```
Warning: Removed 49 rows containing non-finite values (stat_bin).
```

```
Warning: Removed 4 rows containing non-finite values (stat_bin).
```



It is important to note the difference in frequencies between the three histograms displayed above. The OLS heterogeneous treatment effects have a much wider distribution than the Lasso, which then has a wider distribution than the causal forest. Considering this treatment effect (receiving a mailer) should be a positive influence on the outcome_spend, we wouldn't expect to have many observations with CATE < 0. This is where the Causal Forest is far superior to the other two model fits. If there are CATE's below zero this can be interpreted as noise, so the wide distribution on the OLS model means that the predicted CATE isn't very precise.

```
cor_matrix = cor(predict_DT[, c("tau_OLS", "tau_LASSO", "tau_cforest"),
    with = FALSE])

kable(cor_matrix)
```

	tau_OLS	tau_LASSO	tau_cforest
tau_OLS	1.0000000	0.6778590	0.5046731
tau_LASSO	0.6778590	1.0000000	0.7115297
tau_cforest	0.5046731	0.7115297	1.0000000

Looking at the correlation matrix output above, we can see that the OLS and LASSO predicted CATEs are fairly correlated (0.677), whereas the causal forest predicted CATE relative to the OLS CATE only has a correlation of 0.50. The highest correlation between the three models is the causal forest and LASSO's correlation - 0.711.

6.2 Model validation: Lifts

Evaluate the model fit using lift charts and a lift table. The focus is on evaluating how well the models predict the CATE, τ_i. Correspondingly, in a lift chart we create scores based on the estimated (predicted) CATE, and we calculate the average treatment effect in each score (group) based on the observed outcomes in the validation sample.

I recommend to use 20 scores. Also, be careful when calculating the confidence intervals, which correspond to the difference between two sample means.

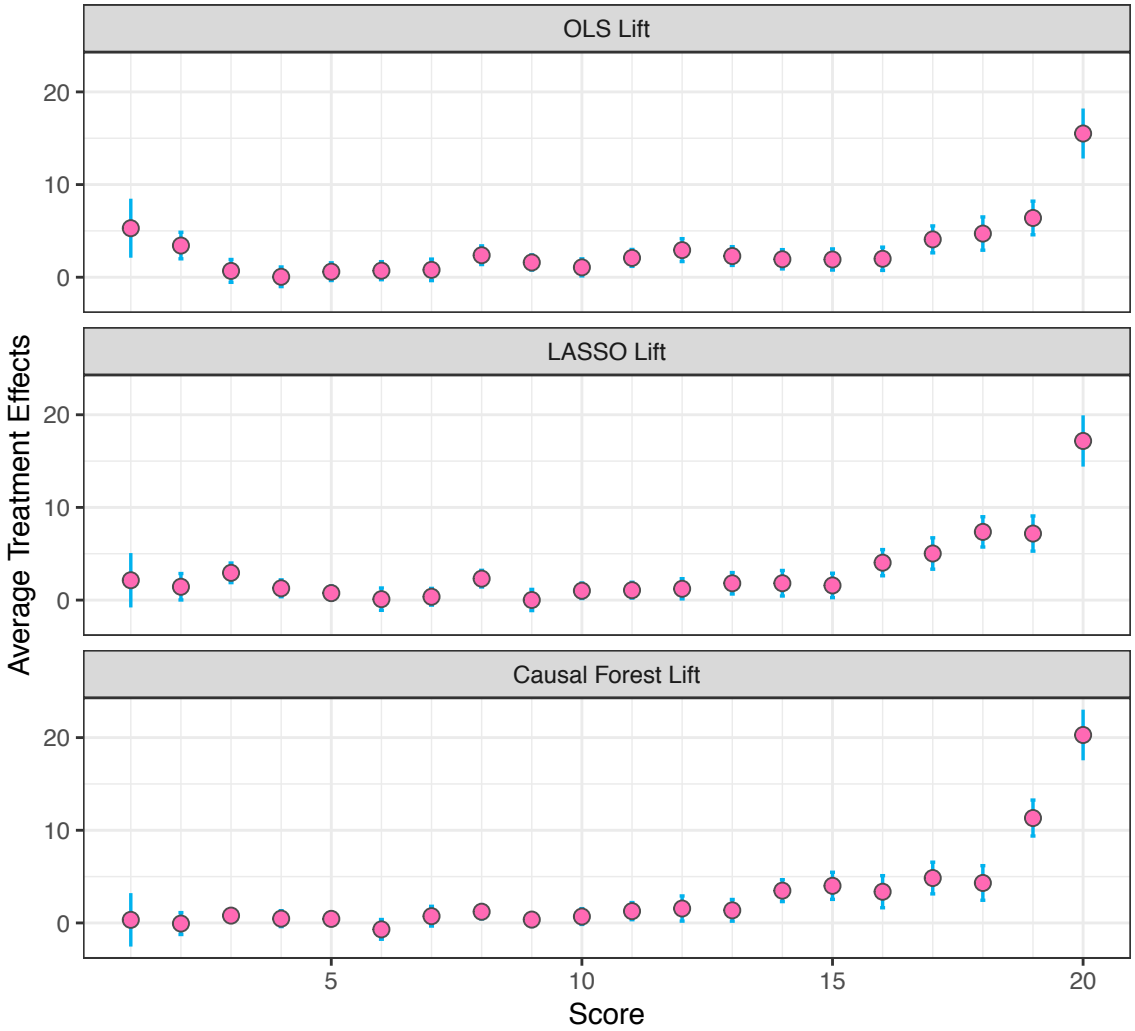
```
### Use the lift table function from assignment 6 and modify it
### so we can calculate lifts given W = 0/1

liftTable <- function(model_name, w, y, score, N_groups = 10) {
    DT = data.table(w = w, y = y, score = score)
    DT[, `:=`(score_group, as.integer(cut_number(score, n = N_groups)))]
    DT[, `:=`(y_1, ifelse(w == 1, y, NA))]
    DT[, `:=`(y_0, ifelse(w == 1, NA, y))]

    lift_DT = DT[, .(model = model_name, score = mean(score),
        y = mean(y_1, na.rm = T) - mean(y_0, na.rm = T), N = .N,
        std_error = sqrt((sd(y_1, na.rm = T)^2)/(length(complete.cases(y_1))) +
            (sd(y_0, na.rm = T)^2)/(length(complete.cases(y_0)))),
        keyby = score_group]

    lift_DT[, `:=`(lower = y + qt(0.025, df = N - 1) * std_error,
        upper = y + qt(0.975, df = N - 1) * std_error)]
    lift_DT[, `:=`(c("std_error", "N"), NULL)]
    lift_DT[, `:=`(lift, 100 * y/mean(y))]

    return(lift_DT)
}

lift_OLS <- liftTable("OLS Lift", predict_DT$W, predict_DT$outcome_spend,
    predict_DT$tau_OLS, 20)

lift_LASSO <- liftTable("LASSO Lift", predict_DT$W, predict_DT$outcome_spend,
    predict_DT$tau_LASSO, 20)

lift_cforest <- liftTable("Causal Forest Lift", predict_DT$W,
    predict_DT$outcome_spend, predict_DT$tau_cforest, 20)

lift_table = rbind.data.frame(lift_OLS, lift_LASSO, lift_cforest)

lift_table[, `:=`(model, factor(model, levels = c("OLS Lift",
    "LASSO Lift", "Causal Forest Lift")))]

kable(lift_table)
```

score_group	model	score	y	lower	upper	lift
1	OLS Lift	-15.7213014	5.2904181	2.1067483	8.4740880	175.1807312
2	OLS Lift	-4.6830450	3.4186614	1.9870200	4.8503029	113.2015642
3	OLS Lift	-2.5176765	0.6813647	-0.5487175	1.9114468	22.5619140
4	OLS Lift	-1.3279078	0.0467584	-1.0120873	1.1056041	1.5483029
5	OLS Lift	-0.5226522	0.5988138	-0.3547363	1.5523640	19.8284229

score_group	model	score	y	lower	upper	lift
6	OLS Lift	0.0920246	0.7052523	-0.2664506	1.6769552	23.3529016
7	OLS Lift	0.6148757	0.7953085	-0.3637442	1.9543612	26.3349167
8	OLS Lift	1.0890744	2.3739914	1.3683841	3.3795988	78.6095818
9	OLS Lift	1.5449817	1.5923768	0.7743675	2.4103861	52.7281060
10	OLS Lift	2.0093216	1.0642888	0.1456547	1.9829229	35.2416180
11	OLS Lift	2.4844828	2.0840474	1.1744019	2.9936929	69.0087126
12	OLS Lift	2.9959787	2.9256596	1.6900775	4.1612418	96.8768780
13	OLS Lift	3.5585476	2.2851666	1.2731098	3.2972235	75.6683407
14	OLS Lift	4.1980175	1.9380858	0.9000232	2.9761483	64.1755100
15	OLS Lift	4.9583901	1.9131384	0.7747680	3.0515087	63.3494317
16	OLS Lift	5.8805534	1.9899143	0.7346289	3.2451997	65.8917001
17	OLS Lift	7.0986615	4.0835194	2.6309806	5.5360582	135.2168951
18	OLS Lift	8.8494203	4.7105706	2.9163899	6.5047512	155.9803356
19	OLS Lift	11.8193719	6.3929235	4.5800803	8.2057666	211.6878052
20	OLS Lift	22.2854359	15.5092841	12.8117956	18.2067726	513.5563315
1	LASSO Lift	-3.8835141	2.1393899	-0.7945622	5.0733419	70.6162169
2	LASSO Lift	-0.5823464	1.4370170	0.0120022	2.8620319	47.4325455
3	LASSO Lift	-0.0839295	2.9402296	1.8830602	3.9973990	97.0500492
4	LASSO Lift	0.2029120	1.2709187	0.3604187	2.1814187	41.9500308
5	LASSO Lift	0.4109345	0.7514146	-0.0086547	1.5114839	24.8024256
6	LASSO Lift	0.5945477	0.0999041	-1.1036178	1.3034259	3.2975982
7	LASSO Lift	0.7789719	0.3574916	-0.5455694	1.2605526	11.7999543
8	LASSO Lift	0.9510521	2.3114685	1.4147847	3.2081524	76.2961281
9	LASSO Lift	1.1185480	0.0088236	-1.1315934	1.1492406	0.2912463
10	LASSO Lift	1.2759596	1.0056331	0.1677962	1.8434700	33.1935780
11	LASSO Lift	1.4183779	1.0665745	0.2140069	1.9191420	35.2051100
12	LASSO Lift	1.5787789	1.2193812	0.1303433	2.3084191	40.2488998
13	LASSO Lift	1.8096170	1.8114831	0.6573907	2.9655756	59.7927881
14	LASSO Lift	2.0998759	1.8176190	0.4419258	3.1933121	59.9953174
15	LASSO Lift	2.4905317	1.5796117	0.2632940	2.8959295	52.1392592
16	LASSO Lift	3.0346899	4.0362679	2.6245821	5.4479538	133.2276904
17	LASSO Lift	3.7895881	5.0267136	3.3345341	6.7188931	165.9199673
18	LASSO Lift	4.9457654	7.3589208	5.7254914	8.9923503	242.9006310
19	LASSO Lift	6.9835741	7.1795846	5.2837992	9.0753699	236.9811633
20	LASSO Lift	14.2776905	17.1735801	14.4057886	19.9413716	566.8594006
1	Causal Forest Lift	-0.4452901	0.3330159	-2.5502061	3.2162378	11.0899101
2	Causal Forest Lift	1.0202616	-0.0673792	-1.2458915	1.1111332	-2.2438236
3	Causal Forest Lift	1.1364831	0.7909996	0.1215477	1.4604515	26.3414306
4	Causal Forest Lift	1.2270922	0.4639777	-0.3930848	1.3210403	15.4511294
5	Causal Forest Lift	1.3124523	0.4377541	-0.2636907	1.1391989	14.5778435
6	Causal Forest Lift	1.4014217	-0.6958642	-1.7585553	0.3668269	-23.1732839
7	Causal Forest Lift	1.4988549	0.7286080	-0.3356299	1.7928458	24.2636987
8	Causal Forest Lift	1.5998674	1.2070625	0.4416039	1.9725212	40.1969282
9	Causal Forest Lift	1.7040919	0.3618427	-0.3810787	1.1047642	12.0498865
10	Causal Forest Lift	1.8276015	0.6898924	-0.1594803	1.5392651	22.9744150
11	Causal Forest Lift	1.9780814	1.2724015	0.3513353	2.1934677	42.3728090
12	Causal Forest Lift	2.1595483	1.5555105	0.1984413	2.9125796	51.8007477
13	Causal Forest Lift	2.3786194	1.3590892	0.1907239	2.5274544	45.2596341
14	Causal Forest Lift	2.6461430	3.4849789	2.3053882	4.6645696	116.0548370
15	Causal Forest Lift	2.9870632	4.0063881	2.5464140	5.4663623	133.4185179
16	Causal Forest Lift	3.4270173	3.3664562	1.6350654	5.0978469	112.1078590
17	Causal Forest Lift	4.0167461	4.8461535	3.1372987	6.5550082	161.3839177

score_group	model	score	y	lower	upper	lift
18	Causal Forest Lift	4.8831474	4.3128032	2.4477689	6.1778375	143.6225832
19	Causal Forest Lift	6.3477775	11.3197861	9.3811922	13.2583799	376.9652447
20	Causal Forest Lift	10.7523466	20.2839755	17.5512643	23.0166867	675.4857150

```
N_groups = 20

ggplot(lift_table, aes(x = score_group, y = y)) + geom_errorbar(aes(ymin = lower,
    ymax = upper), color = "deepskyblue2", size = 0.6, width = 0.1) +
    geom_point(shape = 21, color = "gray30", fill = "hotpink",
        size = 2.5) + scale_x_continuous("Score", limits = c(1,
    N_groups), breaks = seq(0, N_groups, 5), minor_breaks = 1:N_groups) +
    scale_y_continuous("Average Treatment Effects", breaks = seq(0,
        60, 10)) + facet_wrap(~model, ncol = 1) + theme_bw()
```



Looking at the lift table and charts above, we see that the treatment effects are quite a bit harder to predict relative to lifts (as mentioned in lecture), which accounts for the wiggliness in the lower score groups, unlike in the last assignment where the trend was up and right for each higher score group. It's also important to note that, as we saw in the CATE distribution charts by model, the OLS does in fact result in much more "noise" on the lower end which just doesn't come true when looking at real data. In other words, for the OLS

lift table, the 1 score group predicted a -15.72 ATE when in reality we arrive at an ATE of 5.29. Looking at the other 2 models the actual ATEs remain quite small for the bottom 13 score groups, with neither model producing an actual ATE higher than 2.94 in these lower score groups. In looking at the higher score groups, we see that all of the models do a decent job of predicting which individuals will be most affected by the mailer, with OLS doing the worst and the Causal Forest performing the best. In the top 6 groups we see summations of ATEs of 36.5, 44.2, and 51.6 for OLS, LASSO, and Causal Forest, respectively. This means that with Causal Forest we will (most likely) end up with higher ROI than for the other models because we don't have to waste mailers on potential customers that will be unaffected by them.

6.3 Profit predictions

The ultimate assessment of the degree of the predictive power of each model is based on profits. Make sure to first review the corresponding discussion in the *Optimal Targeting Decisions and Heterogeneous Treatment Effects* lecture.

In particular, the total (expected) profits for a targeting strategy $d(\boldsymbol{x}_i)$ can be evaluated based on $\hat{\Pi}(d)$:

$$\hat{\Pi}(d) = \sum_{i=1}^{N} \left(\frac{1 - W_i}{1 - e}(1 - d(\boldsymbol{x}_i)) \cdot (mY_i(0)) + \frac{W_i}{e} d(\boldsymbol{x}_i) \cdot (mY_i(1) - c) \right)$$

Provide the intuition to explain why this approach predicts profits in a randomized sample—this is largely to ensure you understand the method.

Expected profits, not lifts or similar measures of model fit, are the ultimate criterion to evaluate targeting strategies. In practice we don't know the CATE and therefore we estimate it using the steps above. While we could run A/B tests in the field to compare the alternative targeting strategies, this is expensive, so we take the approach of using a randomized sample instead. The data that we are given in this assignment is randomized (as we had to prove in assignment 6) in regards to who received treatment (mailing) and who didn't, and therefore we don't need to run A/B tests to arrive at a conclusion. The profit can be calculated using the inverse probability-weighted profit estimator of targeting profits equation above because the profit only depends on those usable observations when the intended targeting assignment $d(x_i)$, coincides (by chance) with the actual treatment, W_i.

The cost of targeting a customer is \$0.99, and the profit margin is 32.5 percent.

Construct **optimal targeting strategies** for the different CATE estimation methods, and evaluate and compare the targeting profits for the different strategies. Then compare the profits to the baseline when none of the customers are targeted and to a blanket mailing strategy where all customers are targeted.

Also, compare the percentage of all customers targeted across the strategies.

Although not strictly necessary, I recommend to normalize (scale) the predicted profits relative to a customer base of 1 million. This makes the profit numbers easier to interpret and also easier to compare across validation samples with different numbers of customers. Note that the data base of the company contains a much larger number of customers than 1 million.

Hint: First create a function `predictProfit` that takes all the necessary data to predict profits as inputs and returns the predicted profits and the percentage of customers targeted as outputs. The `predictProfitTopPercent` function in the "CRM, Machine Learning, and Profit Evaluation" assignment provides a blueprint for such a function.

6.4 Profits from targeting the top n percent of customers

We can also compare the profitability implications of different targeting strategies as follows. Suppose we target the top n percent of customers, based on predicted *incremental* profits. What is the total expected profit level from targeting these n percent of customers?

Predict the corresponding profit curve on a grid with values $n = 0, 0.01, 0.02, \ldots, 1$. Compare the profit curves across the estimation methods.

```
margin = 0.325  # 32.5 percent
cost = 0.99  # 99 cents

top_percent = seq(from = 0, to = 1, by = 0.01)

### grab the predictProfit function from solution in assignment
### 6 and modify it accordingly

predictProfitTopPercent <- function(model_name, top_percent,
    score, W, spend, margin, cost) {

    # Observed profits for treated and untreated units
    profit_0 = margin * spend
    profit_1 = margin * spend - cost

    # Output table
    K = length(top_percent)
    profits_DT = data.table(model_name = model_name, top_percent = top_percent,
        profit = rep(0, K))

    scale_factor = 1000/length(W)

    for (k in 1:K) {
        if (top_percent[k] < 1e-12) {
            threshold = max(score) + 1  # Make sure everyone is included
        } else if (top_percent[k] > 1 - 1e-12) {
            threshold = min(score) - 1  # Make sure nobody is included
        } else {
            threshold = quantile(score, probs = 1 - top_percent[k])
        }

        T = score >= threshold  # Indicator: Is a customer among the top percent?
        N_0 = sum(1 - T)  # Number of customers not among the top percent
        N_1 = sum(T)  # Number of customers among the top percent

        # Now calculate the mean profits for the treated and
        # untreated units
        mean_profit_0 = sum((1 - T) * (1 - W) * profit_0)/sum((1 -
            T) * (1 - W))
        mean_profit_1 = sum(T * W * profit_1)/sum(T * W)

        if (is.nan(mean_profit_0))
            mean_profit_0 = 0
        if (is.nan(mean_profit_1))
            mean_profit_1 = 0

        profits_DT[k, `:=`(profit, scale_factor * (N_1 * mean_profit_1 +
            N_0 * mean_profit_0))]
    }

    return(profits_DT)
```

```
}

### OLS model profits
OLS_profits = predictProfitTopPercent("OLS", top_percent, predict_DT$tau_OLS,
    predict_DT$W, predict_DT$outcome_spend, margin, cost)
OLS_profits

OLS_opt_n_index = OLS_profits[which.max(OLS_profits$profit),
    top_percent]
OLS_opt_n_index
```

```
[1] 0.2
```

```
max(OLS_profits$profit)
```

```
[1] 2193.396
```

```
### LASSO model profits
LASSO_profits = predictProfitTopPercent("LASSO", top_percent,
    predict_DT$tau_LASSO, predict_DT$W, predict_DT$outcome_spend,
    margin, cost)
LASSO_profits

LASSO_opt_n_index = OLS_profits[which.max(LASSO_profits$profit),
    top_percent]
LASSO_opt_n_index
```

```
[1] 0.23
```

```
max(LASSO_profits$profit)
```

```
[1] 2313.59
```

```
### Causal Forest profits
cforest_profits = predictProfitTopPercent("Causal Forest", top_percent,
    predict_DT$tau_cforest, predict_DT$W, predict_DT$outcome_spend,
    margin, cost)
cforest_profits

cforest_opt_n_index = OLS_profits[which.max(cforest_profits$profit),
    top_percent]
cforest_opt_n_index
```

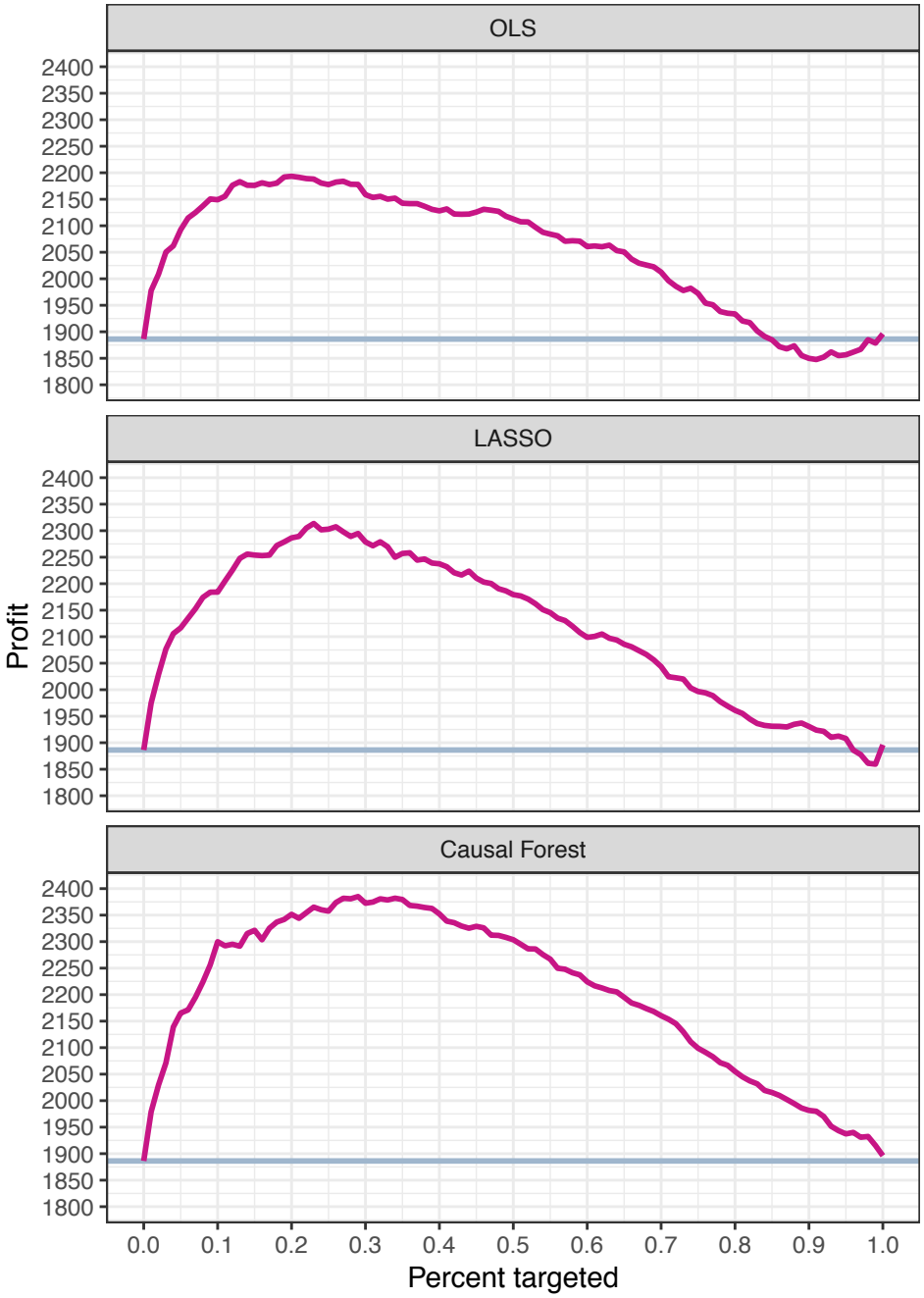
```
[1] 0.29
```

```
max(cforest_profits$profit)
```

```
[1] 2385.037
```

```
##### combine the tables into one profit_table

profit_DT <- rbind.data.frame(OLS_profits, LASSO_profits, cforest_profits)

profit_DT[, `:=`(model_name, factor(model_name, levels = c("OLS",
    "LASSO", "Causal Forest")))]

ggplot(profit_DT, aes(x = top_percent, y = profit)) + geom_hline(data = profit_DT[top_percent ==
    0, .(model_name, profit_0 = profit)], aes(yintercept = profit_0),
```

```
    color = "slategray3", size = 1) + geom_line(color = "mediumvioletred",
    size = 1) + scale_x_continuous("Percent targeted", limits = c(0,
    1), breaks = seq(0, 1, 0.1)) + scale_y_continuous("Profit",
    limits = c(1800, 2400), breaks = seq(1800, 2400, 50)) + theme_bw() +
    facet_wrap(~model_name, nrow = 3)
```



The causal forest performed the best (as expected from our prior analyses), yielding a max profit of $2,385,037 for every 1 million customers, which occurred when target 29% of the customer base. The OLS model performed the worst, yielding a profit of $2,193,396 for every 1 million customers. The OLS model has us targeting only the top 20 percent of the customer base. The LASSO model uses 23 percent of the customer base for mailers and yields a profit of $2,313,590 for every million customers. All three models obviously result in the same profits when targeting 0 percent of the customer base ($1,886,229) and 100 percent of the

customer base ($1,896,120). In looking at the graphs, we see that the curvature of the Causal Forest profit line is flatter at the top as opposed to the other two. This is important because we will be using the best "top n" percent to target customers on out-of-sample data (as in 2018 data) from prior in-sample testing. Because there is a steadier curve at the top of the causal forest rather than a spike like we see in the LASSO graph, we know that there is more margin for error using the causal fit. In fact, only 5 different percentages result in profits greater than 2,300 for the LASSO, whereas 37 different percentages result in profits greater than 2,300 for causal forest.

7 Step 3: How well do the models predict out-of-sample? — Profits and external model validity

Now use the sample of customers from October 2018 to assess how well the model predicts out-of-sample. This is the key test to assess if the model really works. Can we really predict customer behavior one year after the estimation data were collected?

```
# load('/classes/3710501_spr2019/Data/Assignment-7/Randomized-Implementation-Sample-2018.RData')
load("Randomized-Implementation-Sample-2018.RData")
```

Warning: The 2018 data table is also named `crm_DT`, just as the 2017 data.

The approach:

1. Use the model predictions based **only on the 2017 data**.
2. Predict heterogeneous treatment effects for the customers in the October 2018 data.
3. Evaluate the model predictions using the 2018 data.

The structure of the 2018 data is identical to the structure of the 2017 data. In particular, the data contains a randomly selected sample of all customers in the data base, and the treatment assignment W_i is randomized.

Note: Ideally you would perform step 1 by re-estimating all models using *all* of the 2017 data (no training/validation sample split). However, to preserve time, it is perfectly fine if you keep the estimates from the training sample in step 1 to predict the 2018 treatment effects.

```r
# change mailing_indicator to 'W' as we did before
setnames(crm_DT, "mailing_indicator", "W")

# create new predict_DT datatable to store predictions vs.
# actual results
new_predict_DT = crm_DT[, c("customer_id", "outcome_spend", "W"),
    with = FALSE]

####### 1. Use the model predictions based **only on the 2017
####### data**. #######

# fit_OLS

# fit_LASSO

# causal forest given to us

####### 2. Predict heterogeneous treatment effects for the
####### customers in the October 2018 data. #######

### OLS prediction

# need to match colnames in 2018 data to validation_DT
# colnames for prediction

X_2018 = subset(crm_DT, select = names(validation_DT))

X_W_2018 = copy(X_2018)
X_no_W_2018 = copy(X_2018)

X_W_2018[, `:=`(W, 1)]
X_no_W_2018[, `:=`(W, 0)]

predict_W_2018 = predict(fit_OLS, newdata = X_W_2018)
predict_no_W_2018 = predict(fit_OLS, newdata = X_no_W_2018)

# find heterogeneous treatment effect usng OLS
tau_OLS = predict_W_2018 - predict_no_W_2018

### LASSO prediction

# create model.matrix with W = 1
X_with_W = model.matrix(outcome_spend ~ 0 + . * W, data = X_W_2018)
predict_W_LASSO = predict(fit_LASSO, newx = X_with_W, s = "lambda.min")

# create model.matrix with W = 0
X_no_W = model.matrix(outcome_spend ~ 0 + . * W, data = X_no_W_2018)
predict_no_W_LASSO = predict(fit_LASSO, newx = X_no_W, s = "lambda.min")

# find heterogeneous treatment effect usng LASSO
tau_LASSO = predict_W_LASSO - predict_no_W_LASSO

new_predict_DT <- cbind.data.frame(new_predict_DT, tau_OLS, tau_LASSO)
```

```
### Causal prediction - need to merge new_predict_DT with
### predict_DT_2018

new_predict_DT <- merge(new_predict_DT, predict_DT_2018)

setnames(new_predict_DT, "1", "tau_LASSO")

### sanity check on calculations

cor_matrix = cor(new_predict_DT[, c("tau_OLS", "tau_LASSO", "tau_cforest"),
    with = FALSE])

kable(cor_matrix)
```

	tau_OLS	tau_LASSO	tau_cforest
tau_OLS	1.0000000	0.6731906	0.4812304
tau_LASSO	0.6731906	1.0000000	0.6988908
tau_cforest	0.4812304	0.6988908	1.0000000

Above you can see we take the same methodoloy that we used in part 2 of this assignment and apply it to the new 2018 data. We had to remove some of the column names to match the inputs into the predictive functions associated with the different models. Then we switch between applying W=1 and W=0 to all of the data, and then subtract the two results to arrive at the predicted tau or CATE for each model.

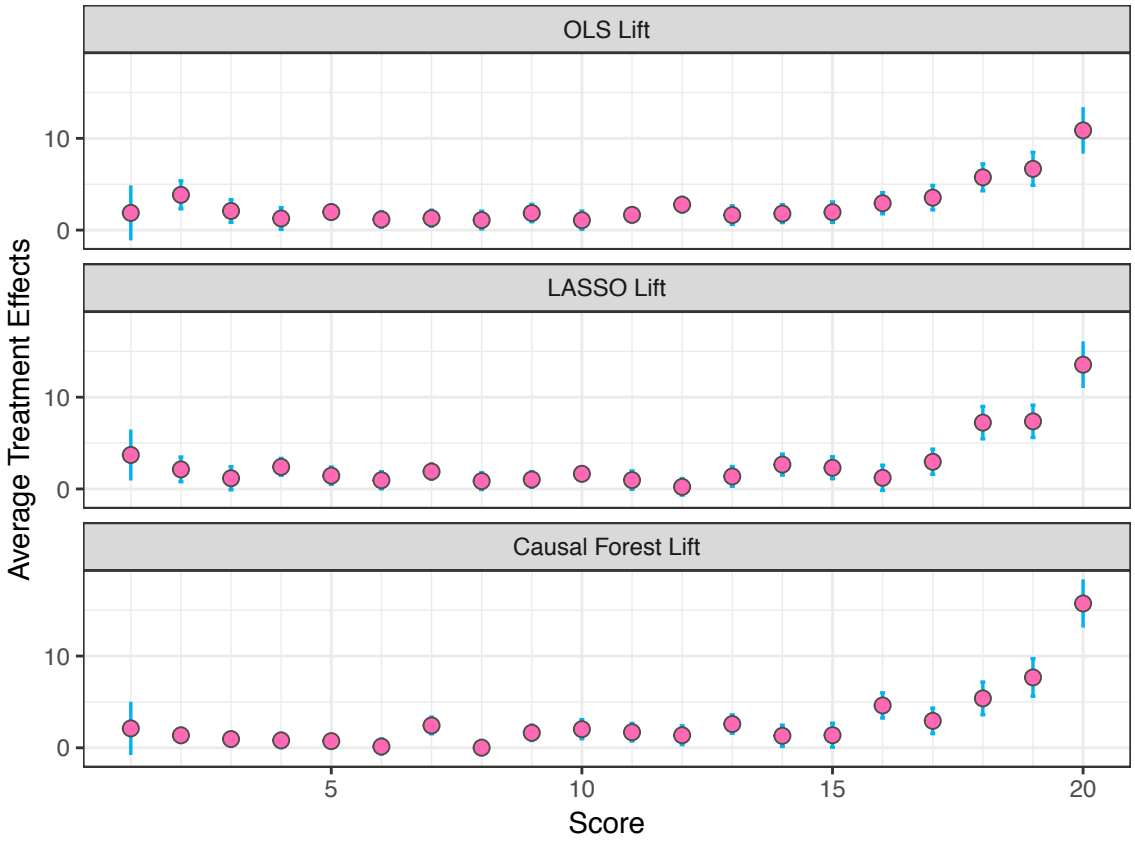
```
######## 3. Evaluate the model predictions using the 2018 data.
######## #######

# create lift tables to examine different models' abilities
# to predict CATE

new_lift_OLS <- liftTable("OLS Lift", new_predict_DT$W, new_predict_DT$outcome_spend,
    new_predict_DT$tau_OLS, 20)

new_lift_LASSO <- liftTable("LASSO Lift", new_predict_DT$W, new_predict_DT$outcome_spend,
    new_predict_DT$tau_LASSO, 20)

new_lift_cforest <- liftTable("Causal Forest Lift", new_predict_DT$W,
    new_predict_DT$outcome_spend, new_predict_DT$tau_cforest,
    20)

new_lift_table = rbind.data.frame(new_lift_OLS, new_lift_LASSO,
    new_lift_cforest)

new_lift_table[, `:=`(model, factor(model, levels = c("OLS Lift",
    "LASSO Lift", "Causal Forest Lift")))]

kable(new_lift_table)
```

score_group	model	score	y	lower	upper	lift
1	OLS Lift	-15.6670190	1.8783428	-1.1232844	4.8799701	65.7637999
2	OLS Lift	-4.7883963	3.8429147	2.3230366	5.3627928	134.5466166
3	OLS Lift	-2.6459224	2.0873975	0.8532579	3.3215371	73.0831404

score_group	model	score	y	lower	upper	lift
4	OLS Lift	-1.4774146	1.2690476	0.0867054	2.4513898	44.4313966
5	OLS Lift	-0.6525607	1.9673940	1.1687440	2.7660441	68.8816265
6	OLS Lift	-0.0048127	1.1658516	0.3220179	2.0096854	40.8183389
7	OLS Lift	0.5305459	1.2957058	0.4198087	2.1716028	45.3647406
8	OLS Lift	1.0065005	1.0937021	0.1260125	2.0613918	38.2922685
9	OLS Lift	1.4740605	1.8584746	0.9095089	2.8074403	65.0681826
10	OLS Lift	1.9543694	1.0765627	0.0813274	2.0717981	37.6921908
11	OLS Lift	2.4571068	1.6591405	0.9186980	2.3995829	58.0891734
12	OLS Lift	2.9959094	2.7743722	1.9778066	3.5709378	97.1352291
13	OLS Lift	3.5917860	1.6357717	0.6265473	2.6449962	57.2709969
14	OLS Lift	4.2693522	1.7992383	0.8230740	2.7754027	62.9942249
15	OLS Lift	5.0704903	1.9518846	0.8420823	3.0616870	68.3386170
16	OLS Lift	6.0721255	2.9285941	1.7748215	4.0823666	102.5347843
17	OLS Lift	7.3757736	3.5402361	2.2102175	4.8702547	123.9493531
18	OLS Lift	9.2501745	5.7517833	4.2839919	7.2195748	201.3791762
19	OLS Lift	12.4016961	6.6769520	4.8825778	8.4713261	233.7708168
20	OLS Lift	23.8133018	10.8705473	8.3366858	13.4044089	380.5953269
1	LASSO Lift	-3.7742753	3.7022694	0.9204902	6.4840486	129.7338601
2	LASSO Lift	-0.5785082	2.1306816	0.7855112	3.4758519	74.6627301
3	LASSO Lift	-0.0660018	1.1677322	-0.0921938	2.4276582	40.9193347
4	LASSO Lift	0.2361313	2.4048672	1.4865345	3.3232000	84.2706650
5	LASSO Lift	0.4538466	1.4483995	0.5083018	2.3884973	50.7544001
6	LASSO Lift	0.6529707	0.9519507	0.0170813	1.8868201	33.3579816
7	LASSO Lift	0.8523139	1.8926403	1.0596730	2.7256075	66.3213562
8	LASSO Lift	1.0349682	0.8590339	-0.0616357	1.7797035	30.1020182
9	LASSO Lift	1.2118572	1.0328322	0.1768389	1.8888255	36.1922073
10	LASSO Lift	1.3805060	1.6501618	0.9921481	2.3081755	57.8244952
11	LASSO Lift	1.5261116	0.9657124	-0.0390409	1.9704656	33.8402154
12	LASSO Lift	1.6973695	0.2273097	-0.6761312	1.1307506	7.9653220
13	LASSO Lift	1.9272426	1.3646766	0.2902623	2.4390908	47.8206043
14	LASSO Lift	2.2229420	2.6564801	1.5162268	3.7967333	93.0876103
15	LASSO Lift	2.6265189	2.3093164	1.1143372	3.5042955	80.9224001
16	LASSO Lift	3.1891340	1.2134489	-0.1574661	2.5843638	42.5213259
17	LASSO Lift	3.9938823	2.9601649	1.5898804	4.3304494	103.7292468
18	LASSO Lift	5.2022763	7.2288543	5.4640324	8.9936762	253.3114335
19	LASSO Lift	7.3287676	7.3655515	5.6072665	9.1238365	258.1015363
20	LASSO Lift	15.0816390	13.5427531	11.0036161	16.0818900	474.5612571
1	Causal Forest Lift	-0.5456512	2.1133955	-0.7858320	5.0126231	74.2981986
2	Causal Forest Lift	0.9940686	1.3574911	0.6000348	2.1149474	47.7237424
3	Causal Forest Lift	1.1262844	0.9531742	0.2850179	1.6213304	33.5096395
4	Causal Forest Lift	1.2181303	0.8063573	0.0347719	1.5779427	28.3481682
5	Causal Forest Lift	1.3045370	0.7283259	0.0416726	1.4149792	25.6049107
6	Causal Forest Lift	1.3923451	0.1395343	-0.6762666	0.9553352	4.9054442
7	Causal Forest Lift	1.4849962	2.4325914	1.5307197	3.3344632	85.5197991
8	Causal Forest Lift	1.5898976	0.0150213	-0.7600209	0.7900635	0.5280848
9	Causal Forest Lift	1.7127072	1.6356216	0.7990503	2.4721929	57.5016549
10	Causal Forest Lift	1.8534753	2.0297511	0.9919364	3.0675658	71.3576083
11	Causal Forest Lift	2.0173259	1.6989592	0.7454745	2.6524440	59.7283409
12	Causal Forest Lift	2.2145411	1.3697314	0.3465048	2.3929580	48.1540605
13	Causal Forest Lift	2.4286579	2.5894487	1.5914426	3.5874549	91.0342491
14	Causal Forest Lift	2.6829684	1.3070189	0.1549642	2.4590735	45.9493485
15	Causal Forest Lift	3.0159899	1.3743358	0.0681664	2.6805051	48.3159308

score_group	model	score	y	lower	upper	lift
16	Causal Forest Lift	3.4650990	4.6150262	3.2444489	5.9856036	162.2451318
17	Causal Forest Lift	4.0673443	2.9357219	1.5452808	4.3261630	103.2077747
18	Causal Forest Lift	4.9442191	5.3869116	3.6044411	7.1693820	189.3814098
19	Causal Forest Lift	6.4385244	7.6703579	5.6252409	9.7154750	269.6578890
20	Causal Forest Lift	10.8138900	15.7307744	13.1056304	18.3559185	553.0286142

```
# graph the lifts for the 2018 data
N_groups = 20

ggplot(new_lift_table, aes(x = score_group, y = y)) + geom_errorbar(aes(ymin = lower,
    ymax = upper), color = "deepskyblue2", size = 0.6, width = 0.1) +
    geom_point(shape = 21, color = "gray30", fill = "hotpink",
        size = 2.5) + scale_x_continuous("Score", limits = c(1,
    N_groups), breaks = seq(0, N_groups, 5), minor_breaks = 1:N_groups) +
    scale_y_continuous("Average Treatment Effects", breaks = seq(0,
        60, 10)) + facet_wrap(~model, ncol = 1) + theme_bw()
```



Another way to evaluate the models is to look at how well we were able to segment to different treatment groups based on their CATE and then compare their scores to the actual measured ATE for each group. As you can see in the plots above, these plots do have similarities to those produced using the validation data in 2017. Causal forest performs the best and OLS performs the worst, but this time the winnner isn't AS CLEAR (still clear just not as notable). Again we see very little separation between the lower score groups, yet once we get to score group 15 we start to notice some separation from the rest of the pack. While we're predicting some negative CATE's we still never see them in the real data. This is why the negative predicted CATE's are typically just representative of noise. It also makes sense that the 2017 models that we created

won't have as strong of predictive abilities on the 2018 data as it did on the 2017 data. This is for a number of reasons, mostly having to do with the fact that the two sets were measured in different slices of time. Macroeconomic factors, competition, and company decisions all can change a comsumer's behavior from one year to the next. That being said, these models are still pretty good at identifying the correct consumers to send mailers to.

While lift charts are beneficial in determining a model's performance, we need to take a look at the profits we would arrive at using the different models. See the analysis below:

```
### use the predictProfit function to examine profits using
### 2018 data

### OLS model profits
OLS_profits = predictProfitTopPercent("OLS", top_percent, new_predict_DT$tau_OLS,
    new_predict_DT$W, new_predict_DT$outcome_spend, margin, cost)
OLS_profits

new_OLS_opt_n_index = OLS_profits[which.max(OLS_profits$profit),
    top_percent]
new_OLS_opt_n_index
```

```
[1] 0.22
```

```
max(OLS_profits$profit)
```

```
[1] 1940.823
```

```
### LASSO model profits
LASSO_profits = predictProfitTopPercent("LASSO", top_percent,
    new_predict_DT$tau_LASSO, new_predict_DT$W, new_predict_DT$outcome_spend,
    margin, cost)
LASSO_profits

new_LASSO_opt_n_index = OLS_profits[which.max(LASSO_profits$profit),
    top_percent]
new_LASSO_opt_n_index
```

```
[1] 0.17
```

```
max(LASSO_profits$profit)
```

```
[1] 1995.019
```

```
### Causal Forest profits
cforest_profits = predictProfitTopPercent("Causal Forest", top_percent,
    new_predict_DT$tau_cforest, new_predict_DT$W, new_predict_DT$outcome_spend,
    margin, cost)
cforest_profits

new_cforest_opt_n_index = OLS_profits[which.max(cforest_profits$profit),
    top_percent]
new_cforest_opt_n_index
```

```
[1] 0.25
```

```
max(cforest_profits$profit)
```

```
[1] 2023.688
```

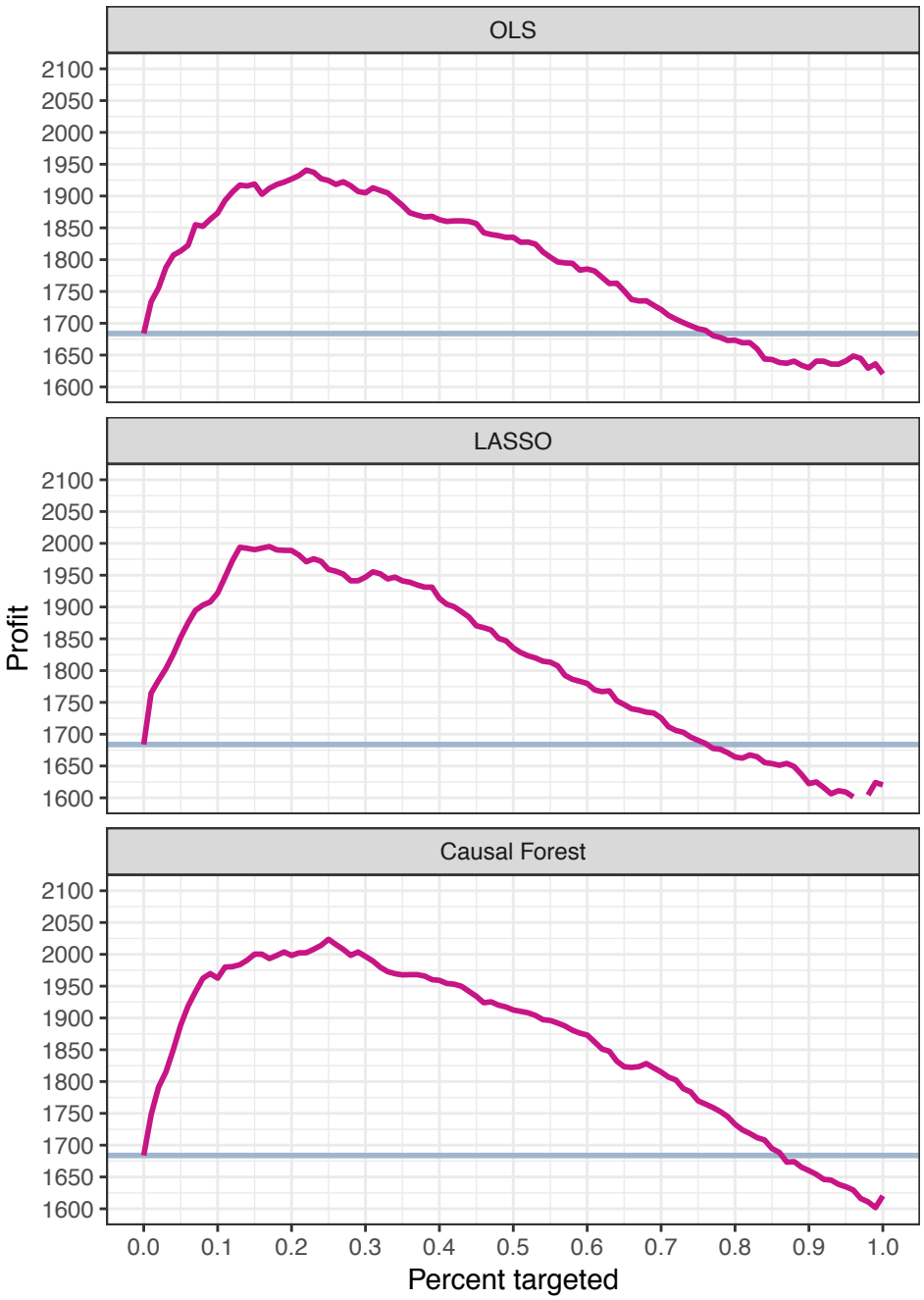
```
##### combine the tables into one profit_table

new_profit_DT <- rbind.data.frame(OLS_profits, LASSO_profits,
    cforest_profits)

new_profit_DT[, `:=`(model_name, factor(model_name, levels = c("OLS",
    "LASSO", "Causal Forest")))]

### find top n = 0, top n = 100 profits
head(new_profit_DT, 1)  # 1683.814
```

```
   model_name top_percent   profit
1:        OLS           0 1683.814
```

```
tail(new_profit_DT, 1)  # 1620.326
```

```
      model_name top_percent   profit
1: Causal Forest           1 1620.326
```

```
ggplot(new_profit_DT, aes(x = top_percent, y = profit)) + geom_hline(data = new_profit_DT[top_percent ==
    0, .(model_name, profit_0 = profit)], aes(yintercept = profit_0),
    color = "slategray3", size = 1) + geom_line(color = "mediumvioletred",
    size = 1) + scale_x_continuous("Percent targeted", limits = c(0,
    1), breaks = seq(0, 1, 0.1)) + scale_y_continuous("Profit",
    limits = c(1600, 2100), breaks = seq(1600, 2100, 50)) + theme_bw() +
    facet_wrap(~model_name, nrow = 3)
```



As you can see all of the models perform worse on the new data, as we were expecting from the analysis above. If we were to backtest the data using the entire range of "top n" percentages of customers, we find that the OLS model would produce a max profit of 1940.83, or $1,940,830 for every million customers, which is down from $2,193,396 using the validation data. The LASSO now only yields a profit of $1,995,019 for every million customers, down from $2,313,590 using the validation data. The causal forest produces a profit of $2,023,688 for every 1 million customers, down from $2,385,037 in the prior problem. This reveals that the difference between the models is not nearly as stark for the out-of-sample / new timeframe data. I imagine this would be a trend we would see in the industry as well.

While I mentioned that the models above do not perform as well as they did on the 2017 data, a lot of that can be overinterpreted. In reality there is still a strong separation between not sending anyone a mailer or sending everyone a mailer and actually targeting customers based on predicted CATE. It's important for the

data scientist to establish a naive vs. targeted performance, which we can do by looking at how well the model performs with a "top n" of 0 and a "top n" of 100. In doing so, we see that we only make $1,683,814 with no mailers, and even less with 100% of customers receiving the mailer ($1,620,326), whereas in 2017 we were making $1,886,229 with 0 percent and $1,896,120 with 100 percent mailers. Our customer base was just less profitable in general in 2018 and therefore it makes our models look worse for the new data. The difference between the baseline (0%) and best causal forest model is $340,000, whereas in 2017 it was $499,000.

That being said, if we were to run a mailing campaign we don't get to test the 100 different "top n" percentages, but instead would use the best percentages from the prior year on the new data. In doing so we arrive at the following profits:

```
#### use prior percentages determined in 2017 data to compare
#### three models profits

#### OLS profit ####

# show old vs. new best 'top percent n'
OLS_opt_n_index
```

```
[1] 0.2
```

```
new_OLS_opt_n_index
```

```
[1] 0.22
```

```
# show profits at the old, predetermined 'best percentage'
OLS_profits[which(OLS_profits$top_percent == OLS_opt_n_index),
    ]
```

```
   model_name top_percent   profit
1:        OLS         0.2 1926.742
```

```
#### LASSO profit ####

# show old vs. new best 'top percent n'
LASSO_opt_n_index
```

```
[1] 0.23
```

```
new_LASSO_opt_n_index
```

```
[1] 0.17
```

```
# show profits at the old, predetermined 'best percentage'
LASSO_profits[which(LASSO_profits$top_percent == LASSO_opt_n_index),
    ]
```

```
   model_name top_percent   profit
1:      LASSO        0.23 1975.767
```

```
#### Causal Forest profit ####

# show old vs. new best 'top percent n'
cforest_opt_n_index
```

```
[1] 0.29
```

```
new_cforest_opt_n_index
```

```
[1] 0.25
```

```
# show profits at the old, predetermined 'best percentage'
cforest_profits[which(cforest_profits$top_percent == cforest_opt_n_index),
    ]
```

```
      model_name top_percent   profit
1: Causal Forest        0.29 2003.714
```

It's a positive sign that the best percentages fore each model between the two sets of data (2017 validation and 2018) are pretty close together - this indicates the the process is repeatable and still useful on out-of-scope data. Using the prior year's "top n" for each model, we see profits of $1,926,742 for the OLS fit, $1,975,767 for the LASSO fit, and $2,003,714 for the causal forest, which is pretty close to the actual optimal profits we arrived at in the prior section.

In conclusion, though the models don't perform as well on the 2018 data as on the 2017 validation data, we shouldn't expect them to. What's important is that these models are still extremely useful in identifying which customers to target based on their predicted reactions to receiving a mailer, which helps to improve the company's ROI - the ultimate goal in taking on this type of project.

8 Optional analysis (bonus)

Do not try any of this before you have thoroughly worked through steps 1-3!

1. Can we improve over the LASSO using an elastic net? Is the elastic net competitive with a causal forest?
2. Suppose you do not have the ability or knowledge to predict heterogeneous treatment effects (currently, this is still true for most firms that use marketing analytics). Instead, you score customers based on the expected level of spending—the traditional CRM approach that you used in the previous assignment. How does targeting using the traditional CRM approach compare to targeting using strategies that are constructed from estimates of the conditional average treatment effects (*incremental* spending)?