Predicting Book Purchases

Rush, Daniel

January 14, 2018

Loading Libraries and data

```
library(dplyr)
library(lubridate)
library(ggplot2)
library(caret)
library(pROC)
library(glmnet)

orders <- read.csv("orders.csv")
booktrain <- read.csv("booktrain.csv")
```

Orders

```
dim(orders)
```

```
## [1] 627955      6
```

```
head(orders)
```

```
##    id   orddate ordnum category qty     price
## 1 914 02DEC2009 314037       20   1  9.203247
## 2 914 02DEC2009 314037       20   1 10.200272
## 3 914 14DEC2010 499719       36   1 10.174706
## 4 914 14DEC2010 499719       20   1 10.200272
## 5 914 14DEC2010 499719       31   1  6.135502
## 6 914 14DEC2010 499719       12   1  8.589699
```

```
str(orders)
```

```
## 'data.frame':    627955 obs. of  6 variables:
##  $ id      : int  914 914 914 914 914 914 914 914 914 914 ...
##  $ orddate : Factor w/ 1863 levels "01APR2008","01APR2012",..: 68 68 815 815 815 815 815 427 427 427
##  $ ordnum  : int  314037 314037 499719 499719 499719 499719 499719 638467 638467 638467 ...
##  $ category: int  20 20 36 20 31 12 20 31 20 20 ...
##  $ qty     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ price   : num  9.2 10.2 10.17 10.2 6.14 ...
```

Booktrain

```
dim(booktrain)
```

```
## [1] 8311    2
```

```
head(booktrain)
```

```
##     id logtarg
## 1 2062       0
## 2 2232       0
## 3 2623       0
## 4 3000       0
## 5 4693       0
## 6 5010       0
```

```r
str(booktrain)
```

```
## 'data.frame':    8311 obs. of  2 variables:
##  $ id     : int  2062 2232 2623 3000 4693 5010 5533 6130 6653 6831 ...
##  $ logtarg: num  0 0 0 0 0 0 0 0 0 0 ...
```

Data Preprocessing

Removing orders with category 99 and price 0

```r
orders_no99 <- orders %>% filter(category != '99') %>% filter(price != 0)
write.csv(orders_no99, "orders_no99.csv", row.names = FALSE)

orders <- orders_no99
```

```r
orders <- mutate(orders, orddate = dmy(orddate), category = factor(category))
```

Partitioning into training and test sets

```r
length(orders$id)
```

```
## [1] 604448
```

```r
length(unique(orders$id))
```

```
## [1] 32790
```

```r
length(booktrain$id)
```

```
## [1] 8311
```

```r
length(unique(booktrain$id))
```

```
## [1] 8311
```

The id's in `booktrain.csv` are all unique. This is not the case with `orders.csv`.

`orders.csv` has 33,355 unique ids. We need to partition the testing and training sets from `orders.csv`.

```r
train <- filter(orders, id %in% booktrain$id)
test <- filter(orders, !(id %in% booktrain$id))
```

Basic EDA

Looking at the number of unique ids in `train` and `test`

```r
length(unique(train$id))
```

```
## [1] 8072
```

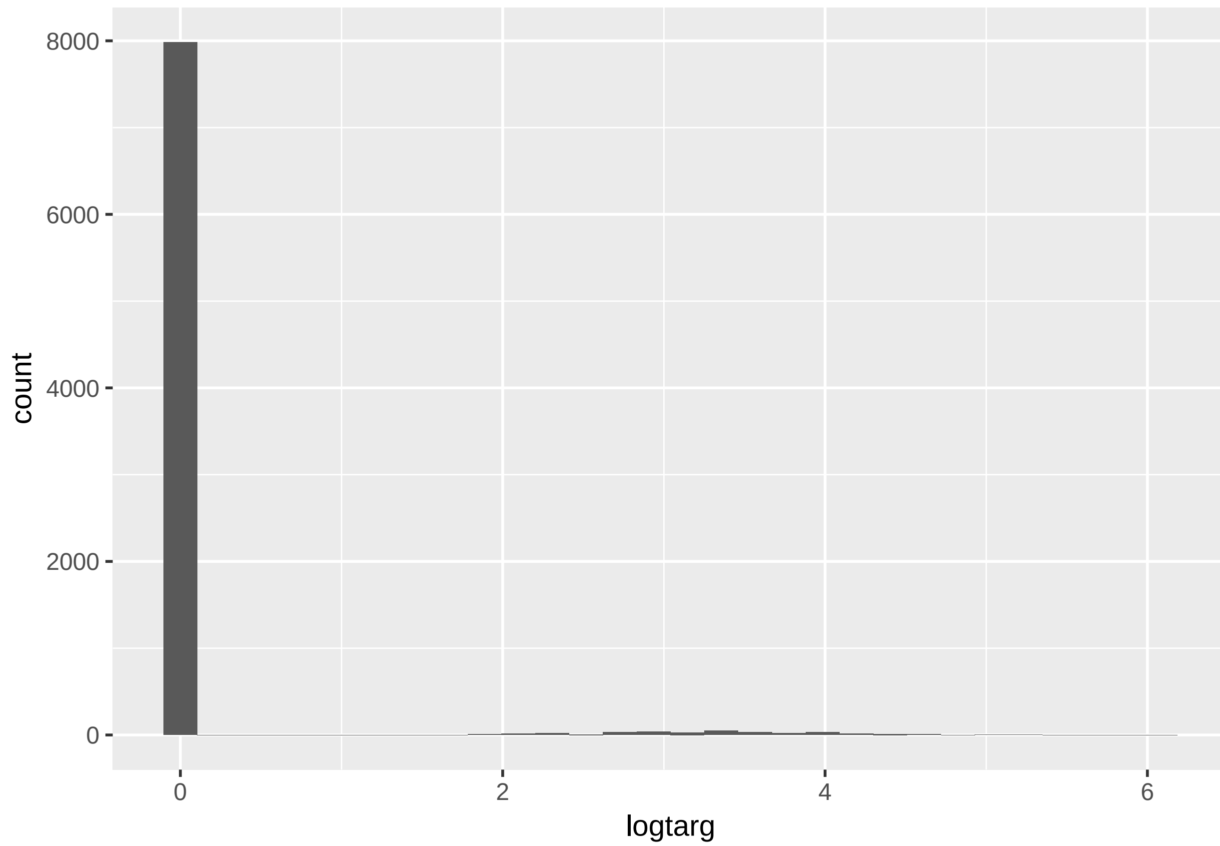
```r
length(unique(test$id))
```

```
## [1] 24718
```

There are 87 ids in `booktrain` that are not present in `orders`.

```r
qplot(data = booktrain, x = logtarg)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



How many customers responded?

```r
length(which(booktrain$logtarg>0))
```

```
## [1] 326
```

Number of orders per customer in descending order.

```r
group_by(orders, id) %>% summarise(n = n_distinct(ordnum)) %>% arrange(desc(n))
```

```
## # A tibble: 32,790 x 2
##          id     n
##       <int> <int>
##  1 13729756   116
##  2  2901870   110
##  3  5361850   107
```

```
## 4 4225988 98
## 5 7219121 90
## 6 4417615 87
## 7 7954352 85
## 8 4456262 83
## 9 3938700 78
## 10 4510020 76
## # ... with 32,780 more rows
```

Feature creation

Creating Aggregated Features

Creating these features in particular, number of orders and total sales per customer in the current year (2014), last year (2013), two years ago (2012), and total before that.

```
this_year <- 2014
last_year <- 2013
ago2_year <- 2012

slstyr_df <- orders %>%
    group_by(id) %>%
    summarise(slstyr = sum(ifelse(year(orddate)==this_year, price*qty, 0)))

ordtyr_df <- orders %>%
    group_by(id) %>%
    summarise(ordtyr = sum(ifelse(year(orddate)==this_year, qty, 0)))

slslyr_df <- orders %>%
    group_by(id) %>%
    summarise(slslyr = sum(ifelse(year(orddate)==last_year, price*qty, 0)))

ordlyr_df <- orders %>%
    group_by(id) %>%
    summarise(ordlyr = sum(ifelse(year(orddate)==last_year, qty, 0)))


sls2ago_df <- orders %>%
    group_by(id) %>%
    summarise(sls2ago = sum(ifelse(year(orddate)==ago2_year, price*qty, 0)))

ord2ago_df <- orders %>%
    group_by(id) %>%
    summarise(ord2ago = sum(ifelse(year(orddate)==ago2_year, qty, 0)))


slsbfr_df <- orders %>%
    group_by(id) %>%
    summarise(slsbfr = sum(ifelse(year(orddate)<ago2_year, price*qty, 0)))

ordbfr_df <- orders %>%
    group_by(id) %>%
    summarise(ordbfr = sum(ifelse(year(orddate)<ago2_year, qty, 0)))
```

```r
# total number of orders per id

tot_orders_df <- group_by(orders, id) %>% summarise(tot_orders =
                         n_distinct(ordnum))

# number of items purchased per order

avg_items_df <- group_by(orders, id) %>% summarise(avg_items = sum(qty)) %>%
    mutate(avg_items = avg_items/tot_orders_df$tot_orders)

# average price per order

avg_price_df <- group_by(orders, id) %>% summarise(avg_price = sum(price*qty)) %>%
    mutate(avg_price = avg_price/tot_orders_df$tot_orders)


# total number of categories purchased

tot_cat_df <- group_by(orders, id) %>% summarise(tot_cat = n_distinct(category))
```

Creating temporal features

```r
# first purchase date

first_pur_df = group_by(orders, id) %>% summarise(first_pur = min(orddate))

# last purchase date

last_pur_df = group_by(orders, id) %>% summarise(last_pur = max(orddate))

# total price of last purchase date

last_tot_price_df = group_by(orders, id, orddate) %>%
    summarise(last_tot_price = sum(price*qty)) %>%
    group_by(id) %>%
    filter(orddate == max(orddate)) %>%
    select(id, last_tot_price)
```

Merging features into a dataframe

```r
# merging all features into temp df

temp_df <- Reduce(function(x, y) merge(x, y, by = "id"), list(tot_orders_df, avg_items_df, avg_price_df

# creating the following features
# 1. the average time between orders
# 2. activity defined as (lifetime - recency)/lifetime, which is the proportion of lifetime a customer
# 3. last order weighted by price

temp_df <- temp_df %>%
    mutate(pur_time_avg = as.integer(last_pur - first_pur)/tot_orders) %>%
```

```
    mutate(activity = as.integer(last_pur - first_pur)/
               as.integer(as.Date("2014-08-01") - first_pur)) %>%
    mutate(last_ord_wt = as.integer(as.Date("2014-08-01") - last_pur)*
               last_tot_price)

# removing last_tot_price now that we have created last_ord_wt

temp_df <- select(temp_df, -last_tot_price)

temp_df <- mutate(temp_df, last_pur = as.integer(last_pur),
                  first_pur = as.integer(first_pur))
```

Adding back the naive features

```
naive_features <- group_by(orders, id) %>%
    summarise(tot_price = sum(price*qty), tot_qty = sum(qty))


temp_df <- merge(temp_df, naive_features, by = 'id')
```

Partition into train and test sets and merge with response

```
train_adv <- filter(temp_df, id %in% booktrain$id)
test_adv <- filter(temp_df, !(id %in% booktrain$id))


merged_train_adv <- merge(train_adv, booktrain, by = 'id')
write.csv(merged_train_adv, "merged_train_adv.csv", row.names = FALSE)
write.csv(test_adv, "test_adv.csv", row.names = FALSE)
```

Training different Models

Now that we have created several features, we will train several models to see which ones work the best.

Simple Linear Regression

```
set.seed(2018-01-20)
tctrl <- trainControl(method = "cv", number = 10)
linear <- train(logtarg ~ .-id-tot_price-tot_qty, data = merged_train_adv, method = "lm", trControl = t
linear
```

```
## Linear Regression
##
## 8072 samples
##   20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
```

```
## Summary of sample sizes: 7265, 7265, 7264, 7265, 7265, 7264, ...
## Resampling results:
##
##   RMSE       Rsquared
##   0.6124811  0.01583755
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

RMSE = 0.6124811

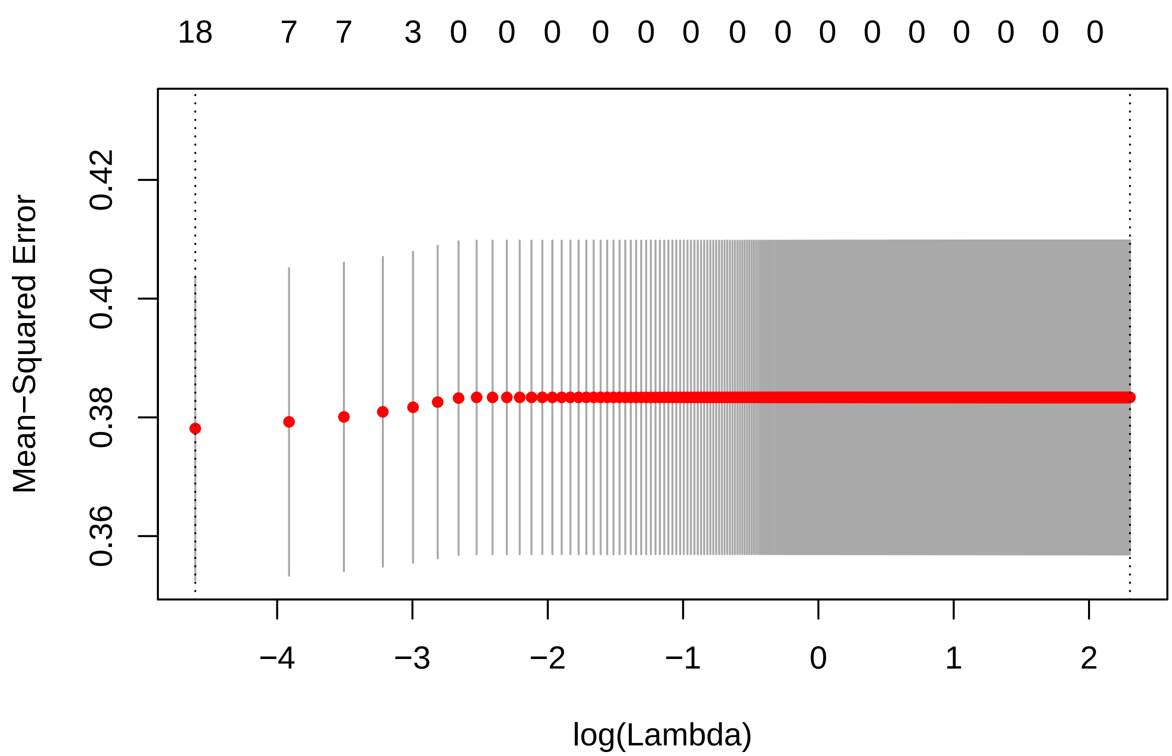
Linear Regression with all possible interactions

Choosing the significant interactions among all possible with lasso

```r
all_mat <- model.matrix(logtarg ~ (.-id)^2-1, data = merged_train_adv)

lassocv <- cv.glmnet(all_mat, merged_train_adv$logtarg, alpha=1,
                     nfold=10, lambda=seq(0,10,0.01))

lambdalasso <- lassocv$lambda.min

print(lambdalasso)
```

```
## [1] 0.01
```

```r
plot(lassocv)
```



```r
small.lambda.index <- which(lassocv$lambda == lassocv$lambda.min)
small.lambda.betas <- coef(lassocv$glmnet.fit)[,small.lambda.index]
# print(small.lambda.betas)


# names(which(small.lambda.betas==0))
```

Lasso doesnt give good results. It fails to remove any of the predictors.

Choosing the significant interactions among all possible with forward stepwise regression

```r
linear_base <- lm(logtarg ~ 1, data = merged_train_adv)
biggest <- formula(lm(logtarg ~ (.-id)^2, data = merged_train_adv))
linear_fwd <- step(linear_base, direction = "both", scope = biggest)
```

Removing the most non significant predictor `activity` and doing backwards again,

```r
# removing activity and creating the new formula to pass to the model
frm <- as.formula(paste0(Reduce(paste0, deparse(formula(linear_fwd))), " - activity - slstyr - last_pur

linear_step_refined <- step(lm(frm, data = merged_train_adv), direction = "both")
```

```
## Start:  AIC=-8001.32
## logtarg ~ slstyr + activity + first_pur + ordtyr + tot_orders +
##     slslyr + ordlyr + tot_cat + activity:ordtyr + first_pur:ordtyr +
##     slstyr:ordtyr + activity:tot_orders + first_pur:slslyr +
##     activity:slslyr + ordtyr:tot_cat + ordlyr:tot_cat + slstyr:activity -
##     activity - slstyr - last_pur - ordlyr
##
##                       Df Sum of Sq    RSS     AIC
## <none>                             2984.5 -8001.3
## - first_pur:slslyr     1     1.205 2985.7 -8000.1
## - slstyr:activity      1     1.627 2986.2 -7998.9
## - activity:slslyr      1     2.631 2987.2 -7996.2
## - ordtyr:tot_cat       1     3.118 2987.6 -7994.9
## - ordlyr:tot_cat       1     4.886 2989.4 -7990.1
## - activity:tot_orders  1     5.153 2989.7 -7989.4
## - slstyr:ordtyr        1     7.415 2991.9 -7983.3
## - activity:ordtyr      1     9.881 2994.4 -7976.6
## - first_pur:ordtyr     1    35.758 3020.3 -7907.2
```

```r
summary(linear_step_refined)
```

```
##
## Call:
## lm(formula = logtarg ~ slstyr + activity + first_pur + ordtyr +
##     tot_orders + slslyr + ordlyr + tot_cat + activity:ordtyr +
##     first_pur:ordtyr + slstyr:ordtyr + activity:tot_orders +
##     first_pur:slslyr + activity:slslyr + ordtyr:tot_cat + ordlyr:tot_cat +
##     slstyr:activity - activity - slstyr - last_pur - ordlyr,
##     data = merged_train_adv)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -1.6593 -0.1286 -0.0815 -0.0586  5.1593
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -3.535e-01  1.827e-01  -1.935 0.053027 .
## first_pur            2.793e-05  1.176e-05   2.374 0.017604 *
## ordtyr              -3.112e-01  3.182e-02  -9.780  < 2e-16 ***
## tot_orders          -1.782e-02  6.548e-03  -2.721 0.006523 **
## slslyr              -4.445e-03  2.526e-03  -1.760 0.078457 .
## tot_cat              5.114e-03  2.580e-03   1.982 0.047500 *
## activity:ordtyr      3.121e-02  6.044e-03   5.165 2.47e-07 ***
## first_pur:ordtyr     1.892e-05  1.925e-06   9.825  < 2e-16 ***
## slstyr:ordtyr        1.484e-05  3.316e-06   4.474 7.78e-06 ***
## activity:tot_orders  2.546e-02  6.825e-03   3.730 0.000193 ***
## first_pur:slslyr     2.774e-07  1.538e-07   1.804 0.071291 .
## activity:slslyr      1.256e-03  4.711e-04   2.665 0.007709 **
## ordtyr:tot_cat       6.782e-04  2.338e-04   2.901 0.003729 **
## ordlyr:tot_cat      -4.983e-04  1.372e-04  -3.632 0.000283 ***
## slstyr:activity     -7.757e-04  3.701e-04  -2.096 0.036107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6086 on 8057 degrees of freedom
## Multiple R-squared:  0.03527,    Adjusted R-squared:  0.0336
## F-statistic: 21.04 on 14 and 8057 DF,  p-value: < 2.2e-16
```

Getting the cross validated RMSE

Using the formula for forward stepwise to get cross validation results

```r
set.seed(2018-01-20)
linear_fwd_cv <- train(formula(linear_step_refined), data = merged_train_adv, method = "lm", trControl =
linear_fwd_cv
```

```
## Linear Regression
##
## 8072 samples
##    9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7265, 7265, 7264, 7265, 7265, 7264, ...
## Resampling results:
##
##   RMSE       Rsquared
##   0.6107102  0.02531327
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

RMSE = 0.6107102

Logistic Regression

Simple logistic regression

```
set.seed(2018-01-20)

merged_train_logistic <- mutate(merged_train_adv,
                                responded = as.factor(ifelse(logtarg>0, 1, 0))) %>%
    select(-logtarg)

levels(merged_train_logistic$responded) <- c("no", "yes")

tctrl_logistic <- trainControl(method = "cv", number = 10, classProbs = TRUE,
                      summaryFunction = twoClassSummary)

logistic <- train(responded ~ .-id-tot_price-tot_qty, data = merged_train_logistic,
                method = "glm", family = binomial, trControl = tctrl_logistic,
                metric = "ROC", maximize = TRUE)
logistic
```

```
## Generalized Linear Model
##
## 8072 samples
##   20 predictor
##    2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7265, 7265, 7265, 7265, 7265, 7264, ...
## Resampling results:
##
##   ROC       Sens       Spec
##   0.673691  0.9998716  0
```

AUC = 0.673691

Logistic with all possible interactions

```
logistic_base <- glm(responded ~ 1, data = merged_train_logistic, family = binomial)

biggest_logistic <- formula(glm(responded ~ (.-id)^2,
                                data = merged_train_logistic, family = binomial))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
logistic_fwd <- step(logistic_base, direction = "both", scope = biggest_logistic)

summary(logistic_fwd)
```

Cross validation on logistic regression

```r
set.seed(2018-01-20)
logistic_fwd_cv <- train(formula(logistic_fwd), data = merged_train_logistic,
                method = "glm", family = binomial, trControl = tctrl_logistic,
                metric = "ROC", maximize = TRUE)
logistic_fwd_cv
```

```
## Generalized Linear Model
##
## 8072 samples
##    5 predictor
##    2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7265, 7265, 7265, 7265, 7265, 7264, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.6813368  0.9998716  0
```

AUC = 0.6813368

This gave a better AUC than before.

Linear model on only `responded = TRUE`

Simple linear model

```r
merged_train_linear <- filter(merged_train_adv, logtarg>0)
set.seed(2018-01-20)
tctrl <- trainControl(method = "cv", number = 10)
linear2 <- train(logtarg ~ .-id-tot_price-tot_qty, data = merged_train_linear, method = "lm", trControl
linear2
```

```
## Linear Regression
##
## 278 samples
##  20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 250, 250, 250, 250, 251, 250, ...
## Resampling results:
##
##   RMSE       Rsquared
##   0.6846061  0.2909004
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Linear model with all interactions

```
linear_base <- lm(logtarg ~ 1, data = merged_train_adv)
biggest <- formula(lm(logtarg ~ (.-id)^2, data = merged_train_linear))
linear_fwd <- step(linear_base, direction = "both", scope = biggest)
summary(linear_fwd)
```

We find that the linear model trained on the subset of the data picks the same features as the one trained on the entire training data.

```
set.seed(2018-01-20)
linear2_fwd_cv <- train(formula(linear_step_refined), data = merged_train_linear,
                        method = "lm", trControl = tctrl)
linear2_fwd_cv
```

```
## Linear Regression
##
## 278 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 250, 250, 250, 250, 251, 250, ...
## Resampling results:
##
##   RMSE       Rsquared
##   0.8333201  0.1711047
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Making predictions

With only linear with all possible interactions

```
pred_linear <- predict(linear_fwd_cv, test_adv)
write.csv(data.frame(id = test_adv$id, yhat = pred_linear),
          "only_linear_regression.csv", row.names = FALSE)
```

With logistic plus linear

```
pred_logistic <- predict(logistic_fwd, test_adv, type = "response")

pred_linear2 <- predict(linear2_fwd_cv, test_adv)

# imposing a hard threshold on the linear model to keep the extreme values in check

pred_linear2 <- ifelse(pred_linear2>20, 20, pred_linear2)

pred_linear2 <- ifelse(pred_linear2<0, 0, pred_linear2)

# generating the final submission file using the linear and logistic model
```

```r
write.csv(data.frame(id = test_adv$id, yhat = pred_logistic * pred_linear2),
          "linear_plus_logistic.csv", row.names = FALSE)
```