MATH/STAT 4450/8456 Machine Learning Competition #1

Will the customer place an order?

The second contest for this course is to construct a model to predict whether a customer will place an order after visiting the website of an online shop.

Image how you will shop online. When you open the website of an online store, it is called you start a *session*. Then you may pick some products and *click* their detail pages. If you like the product, you will add it to the *cart*, and after a few processing steps, you can place the *order*. These behaviors are recorded in the data. If the customer had signed up in the past, then some customer information like purchase history, can be provided as well.

The training data set records the "historical" transactions from Friday morning to Sunday afternoon, and you need to predict the customer's behavior from Sunday evening to Monday morning.

Kaggle website

The data can be downloaded from the kaggle site and you will have to participate the competition through the link https://www.kaggle.com/t/819d387f60194d83b082ab1f9055e872 as it is not open to public.

The maximum daily submission number is 10. So you will need to wait until the next UTC day after submitting 10 results.

Description of variables

- sessionID: Session ID. 1-50000 in `train.csv`, 50001-55111 in `test.csv`.
- hour: Hour when the session was started. Numbers between 0 and 23.
- weekday: Day of the week when the session was started. 5: Friday, 6: Saturday, 7: Sunday, 1: Monday.
- duration: Time in seconds passed since the start of the session.
- clickCount: Number of products that were visited (clicked).
- clickMin: Lowest price of the products visited (clicked).
- clickMax: Highest price of the products visited (clicked).
- clickTotal: Sum of the prices of all the products visited (clicked).
- cartCount: Number of products that were added to the cart.
- cartMin: Lowest price of the products in the cart.
- cartMax: Highest price of the products in the cart.
- cartTotal: Sum of the prices of all the products in the cart.
- cartStep: Purchase processing step. Possible values: 1,2,3,4,5.
- status: Whether the customer is online. 'y': yes. 'n': no.
- availability: Whether the cart is orderable or not. There are seven possible values.
- customerID: customer ID. 1-25038 in `train.csv`. In `test.csv` there exist both old and new customers.
- purchase: Highest purchase price in history for the customer.
- score: Customer score evaluated by the online store.
- account: Lifetime of the customer's account in months.
- payments: Number of payments made by the customer.
- age: Age of the customer.
- salutation: Salutation of the customer. 1: Mr, 2: Ms, 3: Company.
- lastOrder: Time in days passed since the last order.
- order: (Response variable) Whether the order is placed. 'y': yes. 'n': no.

Data

Here is a quick look at the data.

```
train = read.csv("data/train.csv")
str(train)
```

```
## 'data.frame':    429013 obs. of  24 variables:
##  $ sessionID   : int  1 1 1 2 2 2 2 2 3 3 ...
##  $ hour        : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ weekday     : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ duration    : num  0 11.9 39.9 0 15.6 ...
##  $ clickCount  : int  1 1 1 0 0 0 0 0 9 11 ...
##  $ clickMin    : num  60 60 60 NA NA ...
##  $ clickMax    : num  60 60 60 NA NA ...
##  $ clickTotal  : num  60 60 60 NA NA ...
##  $ cartCount   : int  1 1 1 0 0 0 0 0 1 2 ...
##  $ cartMin     : num  60 60 60 NA NA ...
##  $ cartMax     : num  60 60 60 NA NA ...
##  $ cartTotal   : num  60 60 60 NA NA ...
##  $ cartStep    : int  NA 2 NA 2 NA 4 4 NA NA NA ...
##  $ status      : Factor w/ 2 levels "n","y": NA 2 2 2 2 2 2 2 NA NA ...
##  $ availability: Factor w/ 7 levels "completely not determinable",..: NA 3 3 3 3 3 3 3 NA NA ...
##  $ customerID  : int  1 1 1 NA NA NA NA NA 3 3 ...
##  $ purchase    : int  600 600 600 NA NA NA NA NA 1800 1800 ...
##  $ score       : int  70 70 70 NA NA NA NA NA 475 475 ...
##  $ account     : int  21 21 21 NA NA NA NA NA 302 302 ...
##  $ payments    : int  1 1 1 NA NA NA NA NA 12 12 ...
##  $ age         : int  43 43 43 NA NA NA NA NA 45 45 ...
##  $ salutation  : int  1 1 1 NA NA NA NA NA 1 1 ...
##  $ lastOrder   : int  49 49 49 NA NA NA NA NA 11 11 ...
##  $ order       : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 2 2 2 2 ...
```

```
table(train$order)
```

```
##
##      n      y
## 138983 290030
```

It seems that there are more placed orders than non-placed orders. However, if using the sessionID, you'll find that there are a little more non-placed orders than placed orders.

```
table(train$order[!duplicated(train$sessionID)])
```

```
##
##     n     y
## 26822 23178
```

In the test data, there are 5111 sessions that you need to predict.

```
test = read.csv("data/test.csv")
str(test)
```

```
## 'data.frame':    45068 obs. of  23 variables:
##  $ sessionID   : int  50001 50001 50001 50001 50001 50001 50001 50001 50001 50002 ...
##  $ hour        : int  18 18 18 18 18 18 18 18 18 18 ...
##  $ weekday     : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ duration    : num  137 190 343 411 460 ...
##  $ clickCount  : int  3 3 6 8 10 10 11 11 11 7 ...
```

```
##  $ clickMin    : num  40 40 17 17 17 ...
##  $ clickMax    : num  40 40 40 40 40 ...
##  $ clickTotal  : num  80 80 114 150 190 ...
##  $ cartCount   : int  1 1 2 3 4 4 5 5 5 1 ...
##  $ cartMin     : num  40 40 17 17 17 ...
##  $ cartMax     : num  40 40 40 40 40 ...
##  $ cartTotal   : num  40 40 57 75 95 ...
##  $ cartStep    : int  2 NA NA NA NA 1 NA 1 NA NA ...
##  $ status      : Factor w/ 2 levels "n","y": 2 2 NA NA NA 2 NA 2 2 NA ...
##  $ availability: Factor w/ 7 levels "completely not determinable",..: 3 3 NA NA NA 3 NA 3 3 NA ...
##  $ customerID  : int  25039 25039 25039 25039 25039 25039 25039 25039 25039 25040 ...
##  $ purchase    : int  1300 1300 1300 1300 1300 1300 1300 1300 1300 1200 ...
##  $ score       : int  489 489 489 489 489 489 489 489 489 543 ...
##  $ account     : int  188 188 188 188 188 188 188 188 188 43 ...
##  $ payments    : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ age         : int  49 49 49 49 49 49 49 49 49 29 ...
##  $ salutation  : int  1 1 1 1 1 1 1 1 1 2 ...
##  $ lastOrder   : int  65 65 65 65 65 65 65 65 65 184 ...
```

```
summary(unique(test$sessionID))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   50001   51279   52556   52556   53834   55111
```

Task

1. Create the most accurate classifier that you can for the data, as measured by the test data.
2. Write a 5-8 page slides summarizing your approach to
   (a) formulating the model (design) matrix,
   (b) building the classifier, and
   (c) your findings from the data.

Format of submission

Your submission file should be in the csv format with two columns: `sessionID` and `order`. Example:

```
sessionID,order
50001,n
50002,y
50003,n
...
55111,n
```

You can also download the `sample_submission.csv` file from Kaggle and replace the `order` column with your prediction, then submit it to Kaggle.

Deadlines:

- March 17 (11:59 pm): Final prediction submission.
- March 19 (in class): Presentation. 6 minutes per team.
- March 19 (11:59 pm): Slides and code submission.

Grading:

- Total points: 50 (+2)
  - Accuracy of classifier: 20
    - Score = 20 * (accuracy rate)^2
  - Progress made from multiple submissions: 10
    - Number of good submissions (decreasing error rate): 6
    - Amount of decreasing of the error rate: 4
  - Presentation: 12 (+2)
    - Model matrix: 4
    - Model selection: 4
    - Model assessment: 4
    - Team battle: (+2)
  - Met the deadlines: 3
  - Within-team adjustment: 5

Teams

- Undergrad 1: Grace Doan, Matt Pelz, Tao Wu
- Undergrad 2: Judge Hiciano, Kevin Rodenhausen, Consuelo Sobalvarro, Chenggong Zhang
- Grad 1: Mohammad Ali Takallou, Gnapika Talluri, Alexander Way
- Grad 2: Kenton Hummel, Dongqi Lai, Lakshmi Sravani Garimella
- Grad 3: Ali Al-Ramini, Michael Kuhlenengel, Mamadou Traore
- Grad 4: Ahmad Almaghrebi, Mohammad H Hasan, Brian Puckett
- Grad 5: Ali Al-Ghaithi, Kenzie Maschka, Rama Krishna Thelagathoti
- Grad 6: Morgan Foxworthy, Ati Soleimani Javid, Bikram Maharjan
- Grad 7: Ru Ng, Ramin Ziaei Tabari

Presentation

Two undergrad teams will be in a battle group. Seven graduate teams will be assigned into four battle teams based on their final submission results – the battling teams will have very similar error rates. All the audience will vote for the better presenter and one of the two teams who gets higher votes will receive extra points. One of the undergrad teams will not have a competitor in their presentation, so this team need to win 2/3 of the votes to receive the extra points.

Peer evaluation

The within-team adjustment grade will be calculated based on the average evaluation from the team members. Everyone will rate each team member (including yourself) regarding her or his contribution to the team effort on the peer evaluation form. The total team effort includes: leadership, arranging and/or attending meetings, contributing creative ideas, coding, writing, presenting the results, and any other activities you feel are important for the success of the contest.