BA810 Lab 1: Descriptive analytics and regression

The problem

Cogo Labs contacts consumers via email with promotional offers. While consumers have opted-in to receiving email offers, some tend to ignore them. Because email providers like Gmail and Yahoo! use email open rates as an input to their spam detection filters, when Cogo Labs contacts unresponsive customers (i.e., those unlikely to open the emails) it faces the potential cost of having its future messages classified as spam. Therefore, it is crucial for the firm that it learns which customers are unlikely to open emails and avoid contacting them in the first place.

The data

Cogo Labs has provided us with a real-world dataset containing the browsing behavior and email open-rates of approximately 400,000 consumers. The dataset contains the following columns:

1. user_id: a unique user id
2. p_open: fraction of times user opened an email
3. browser1-3: Chrome, Firefox, and Safari (1 if the user was seen using that browser, zero otherwise)
4. device_type1-4: Android, iDevice, other mobile, and desktop (1 if the user was seen using that device, zero otherwise)
5. activity_observations: how many times the user has been seen online
6. activity_days: on how many days has the user been seen online
7. activity_recency: how many days since the last time we saw the user online
8. activity_locations: at how many unique locations have we seen the user
9. activity_ids: on many unique computers (inlc. phones, tablets) have we seen the user
10. state, age, gender: obvious meaning

The lab

Preparing for the lab

Let's load some useful libraries.

```
library(tidyverse)
library(ggplot2)
library(ggthemes)
library(scales)
theme_set(theme_bw())
```

Loading the dataset

Your first task is to load the dataset. You will have to change the path below to match the location of the dataset on your computer.

Summary stats

Here are some basic summary stats for data column. Anything interesting?

```
str(dd)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 288298 obs. of  17 variables:
##  $ state               : chr  "AK" "AK" "AK" "AK" ...
##  $ user_id             : num  1087 1656 2071 2228 2500 ...
```

```
##  $ browser1             : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ browser2             : num  0 1 1 1 0 1 1 1 0 0 ...
##  $ browser3             : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ device_type1         : num  0 0 0 0 0 0 0 1 0 1 ...
##  $ device_type2         : num  0 1 0 0 1 1 0 1 1 0 ...
##  $ device_type3         : num  0 1 0 0 1 1 0 1 1 1 ...
##  $ device_type4         : num  1 1 1 1 0 1 1 1 0 0 ...
##  $ activity_observations: num  74 39 9 14 2 139 13 41 118 5 ...
##  $ activity_days        : num  15 22 8 11 2 49 6 14 11 5 ...
##  $ activity_recency     : num  100 36 27 68 100 71 150 78 37 106 ...
##  $ activity_locations   : num  2 4 3 2 1 3 1 2 2 1 ...
##  $ activity_ids         : num  3 7 2 2 1 8 1 4 3 5 ...
##  $ age                  : num  20 26 28 19 21 27 24 22 28 33 ...
##  $ gender               : chr  "M" "F" "M" "M" ...
##  $ p_open               : num  0 0.0181 0.0359 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   state = col_character(),
##   ..   user_id = col_double(),
##   ..   browser1 = col_double(),
##   ..   browser2 = col_double(),
##   ..   browser3 = col_double(),
##   ..   device_type1 = col_double(),
##   ..   device_type2 = col_double(),
##   ..   device_type3 = col_double(),
##   ..   device_type4 = col_double(),
##   ..   activity_observations = col_double(),
##   ..   activity_days = col_double(),
##   ..   activity_recency = col_double(),
##   ..   activity_locations = col_double(),
##   ..   activity_ids = col_double(),
##   ..   age = col_double(),
##   ..   gender = col_character(),
##   ..   p_open = col_double()
##   .. )
```

Describing the data

Let's start answering some basic questions about the data

1. What fraction of users use each browser? Note that some users use more then one browser.

```
cogo_browsers <- dd %>%
  summarise(
    p_chrome = mean(browser1),
    p_firefox = mean(browser2),
    p_safari = mean(browser3))
```

Let's rearrange the dataset so that each row corresponds to a different browser. `pivot_longer` converts a "wide" dataset to a "long" dataset, where every column becomes a row.

```
cogo_browsers <- cogo_browsers %>%
  pivot_longer(
    c(p_chrome,p_firefox,p_safari),
    names_to="browser",
    values_to="p")
```

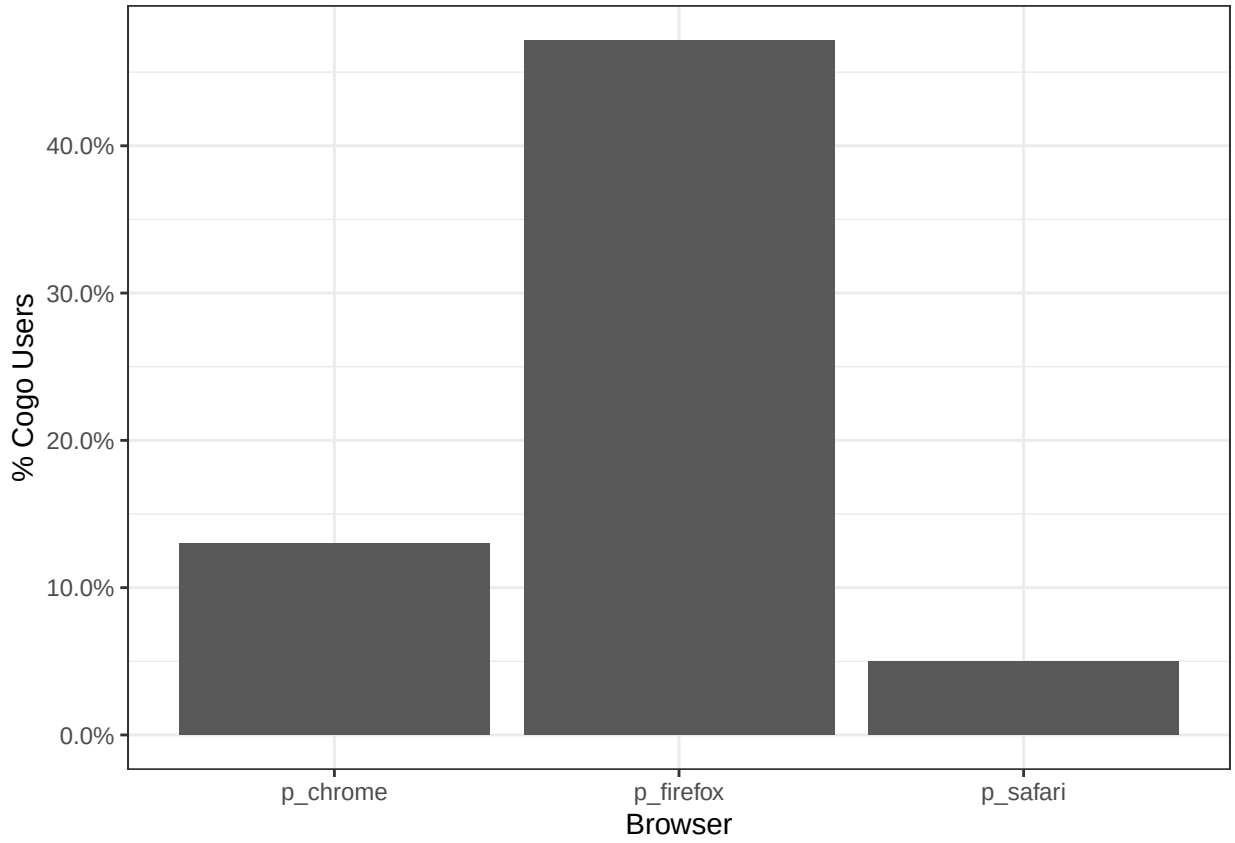
Check the result:

```
cogo_browsers
```

```
## # A tibble: 3 x 2
##   browser        p
##   <chr>      <dbl>
## 1 p_chrome  0.130
## 2 p_firefox 0.472
## 3 p_safari  0.0496
```

Now, let's plot the distribution of browsers using `gglot2`.

```
ggplot(cogo_browsers, aes(browser, p)) +
  geom_bar(stat="identity") +
  scale_x_discrete("Browser") +
  scale_y_continuous("% Cogo Users", label=percent_format())
```

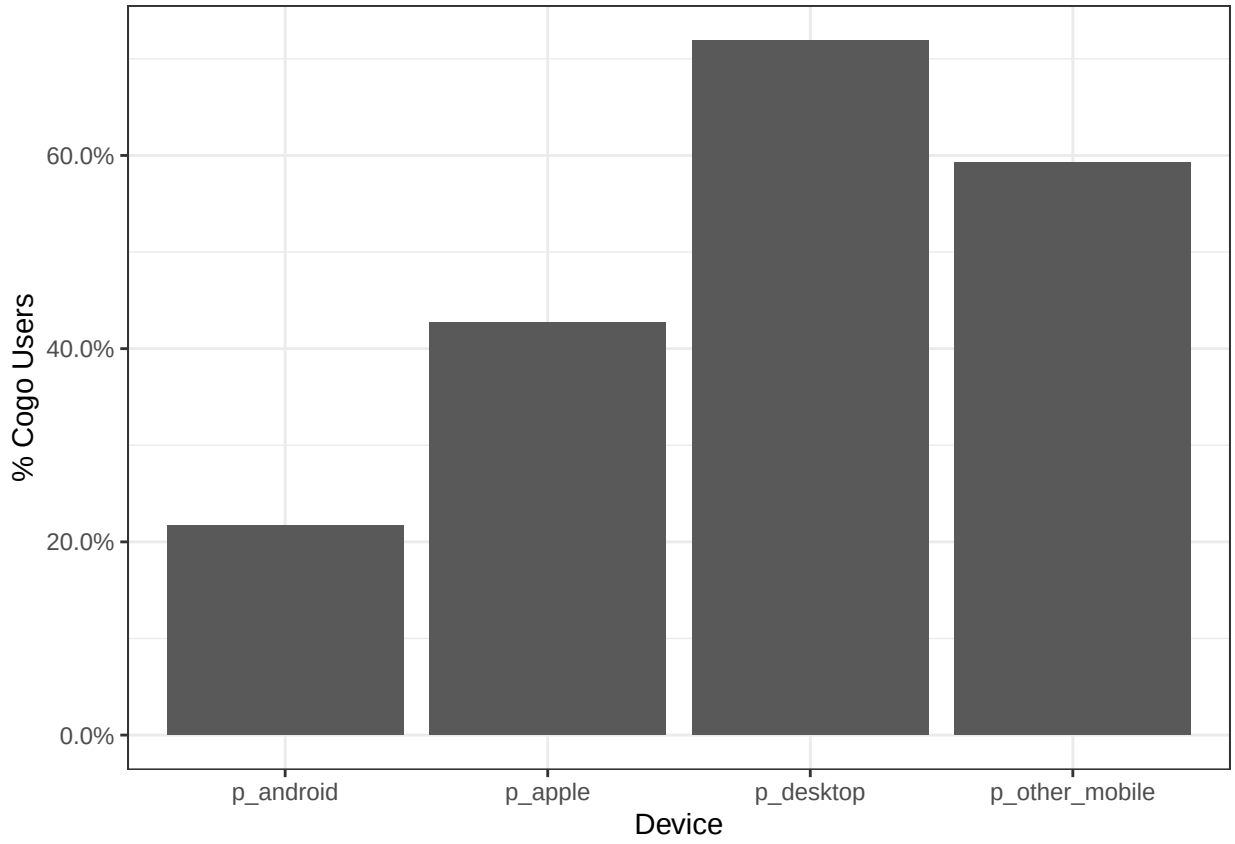


2. What fraction of users use each device? Use code similar to the above to answer this question.

```
cogo_devices <- dd %>%
  summarise(
    p_android = mean(device_type1),
    p_apple = mean(device_type2),
    p_other_mobile = mean(device_type3),
    p_desktop = mean(device_type4)) %>%
  pivot_longer(
    c(p_android,p_apple,p_other_mobile,p_desktop),
    names_to = "device",
```

```
    values_to = "p")

ggplot(cogo_devices, aes(device, p)) +
  geom_bar(stat="identity") +
  scale_x_discrete("Device") +
  scale_y_continuous("% Cogo Users", label=percent_format())
```



3. What fraction of users use more than one browser?

First, we will create a couple of new variables. Can you describe what these variables mean?

```
dd %<>%
  mutate(num_browsers = browser1 + browser2 + browser3) %>%
  mutate(two_or_more_browsers = num_browsers > 1)
```

Now, we are ready to compute the percentage of multi-browser users.

```
dd %>%
  summarise(mean(two_or_more_browsers))
```

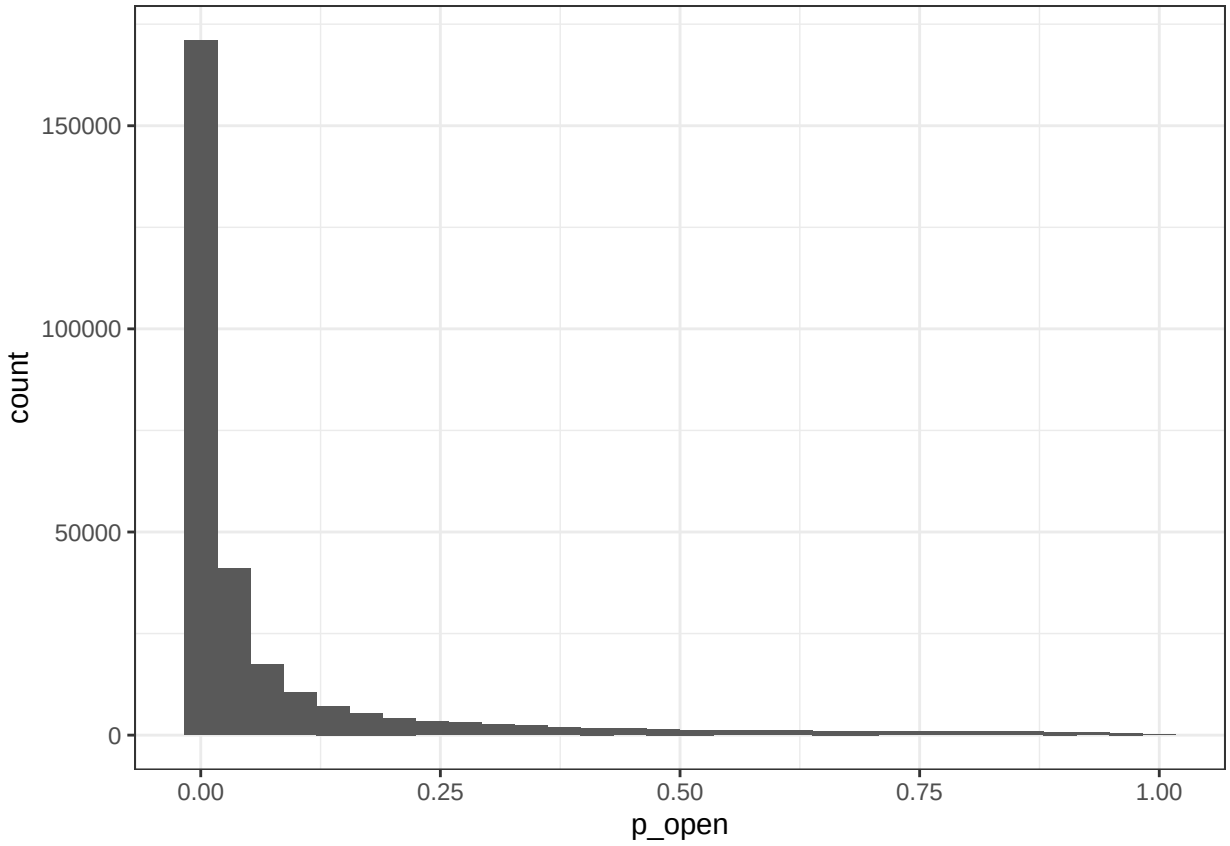
```
## # A tibble: 1 x 1
##   `mean(two_or_more_browsers)`
##                          <dbl>
## 1                       0.0733
```

4. What is the distribution of email open-rates?

Now, we can plot the a histogram of `p_open`.

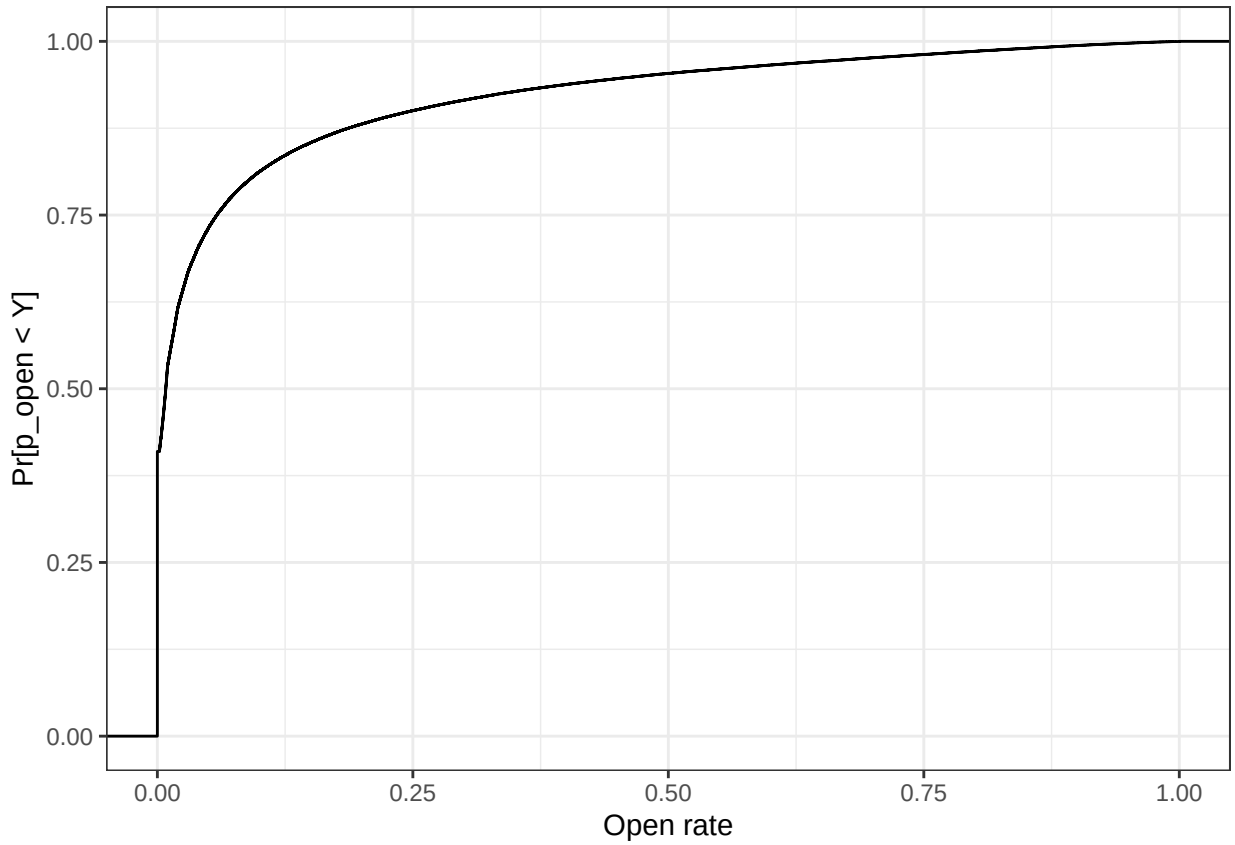
```
ggplot(dd, aes(p_open)) +
  geom_histogram()
```

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



Instead of plotting a histogram, we could have plotted a cumulative distribution.

```
ggplot(dd, aes(p_open)) +
  scale_y_continuous("Pr[p_open < Y]") +
  scale_x_continuous("Open rate") +
  stat_ecdf()
```



Looking at the figure above, approximately what fraction of consumers have an open rate of at 25% or less?

Some extra questions you may want to consider:

1. Are multi-homing users (those using more than one device) more or less likely to open an email?
2. On which device are multi-homing users most likely to open an email?
3. What is the relationship between the various activity indicators and open rates?
4. Using faceting, plot the relationship between age and p_open by gender and/or state.

Regression

You have now explored the computation advertising dataset provided by Cogo Labs, and you have some intution as to the determinants of email open rates. It is now time to translate this intuition into a machine learning to predict email open rates for new customers.

Creating some irrelevant predictors

We are going to be experimenting with how including different predictors impacts our results. We will want some of those predictors to be ones that we know should be irrelevent to our model. Let's add those predictors now. You could add irrelevant predictors by adding random numbers (here `runif` isn't "run if", it's short for "r uniform". You could try other distributions `rnorm`, `rbinom`, or `rpois`. Note also that if we used `1` instead of `n()` we would get the same random number in every row.):

```
#This would add a new column, ran1.
dd %>% mutate(ran1=runif(n(),min=0,max=100))
```

Or by selecting the digit in, say, the tens place in the user ID:

```
dd %>% mutate(tensdig=(as.integer(user_id/10)%%10))
```

Try adding ten irrelevant predictors to the data frame.

```
dd <- dd %>% mutate(
  ir0=..., ir1=..., ir2=..., ir3=..., ir4=...,
  ir5=..., ir6=..., ir7=..., ir8=..., ir9=...)
```

Train and test datasets

We will suffix training-related data with `.train`, and testing-related data with `.test` to easily distinguish them.

Now we are ready to split the data into a test data set and a train data set. We will pseudorandomly select 100,000 of the rows to be in the testing set. It can be useful if we all are generating the same pseudorandom numbers. We can do this by giving R a "seed":

```
set.seed(810)
```

So, for example, now if you try to select a random number from the normal distribution, like so, you will get the same number as below.

```
rnorm(1)
```

```
## [1] 1.04923
```

Now we can split our data into two sets:

```
test_index <- sample(nrow(dd), 100000) # assign 100K random rows to the test set
# now split
dd.test <- dd[test_index,]
dd.train <- dd[-test_index,]
```

Linear regression

Later, you will be asked to experiment with multiple formulas. Instead of overwriting `f1`, define a new `f2` and `f3` etc for each model.

```
# here's one simple formula -- it's up to your to add more predictors as you see fit (haha)
f1 <- as.formula(p_open ~ browser1 + browser2 + browser3)
f2 <- ...
f3 <- ...
...
```

Now, we extract the true response value to later compare our predictions from various models against

```
y.train <- dd.train$p_open
y.test <- dd.test$p_open
```

The simplest thing we can try is linear regression. Linear regression works uses the formula interface.

```
fit.lm1 <- lm(f1, dd.train)
```

Now, let's compute an MSE on the training data.

```
yhat.train.lm1 <- predict(fit.lm1)
mse.train.lm1 <- mean((y.train - yhat.train.lm1)^2)
```

So, we have $\text{MSE}_{Train} = 0.0284975$.

Now, is a good time to recall that OLS minimizes the sum least of squares, i.e., the following objective function: $\text{MSE}_{Train} = \frac{\sum_i^n (\hat{y_i} - y_i)^2}{n}$, where $\hat{y_i} = \beta_0 - \sum_{j=1}^p \beta_j x_{ij}$.

We can also compute the MSE of the test data which is the quantity we really care about:

```
yhat.test.lm1 <- predict(fit.lm1, dd.test)
mse.test.lm1 <- mean((y.test - yhat.test.lm1)^2)
```

We have that $\text{MSE}_{Test} = 0.028651$.

Adding more predictors to the model

Add predictors into the model that you think will improve the model, like `activity_recency`. How does this change MSE_{Train} and MSE_{Test}?

```
f2 <- as.formula(p_open ~ browser1 + browser2 + browser3 + activity_recency + ...)
fit.lm2 <- ...
yhat.train.lm2 <- ...
mse.train.lm2 <- ...
yhat.test.lm2 <- ...
mse.test.lm2 <- ...
```

Training using a small training sample

The training data contains 188298 observations. If we used a very small training sample, say 30, would you expect that to improve our model? We can use the following code to just use a subset of our data for training.

```
dd.train.small.sample.size <- 30
dd.train.small.sample <- dd.train[sample(nrow(dd.train), dd.train.small.sample.size),]
y.train.small.sample <- dd.train.small.sample$p_open
```

Test now to see how that would change MSE_{Train} and MSE_{Test}.

If in addition to using a small sample we also added in predictors we know to be irrelevant, would we expect this to improve our model? Test now to see how that would change MSE_{Train} and MSE_{Test} by adding the ten irrelevant predictors to the model you defined earlier.

How does a model with these irrelevant predictors impact MSE_{Train} and MSE_{Test}? Can you explain why?

Interactions

Sometimes we may think that there is some sort of interaction between our variables. For example perhaps we suspect that college-aged users who have a low number of activity locations behave similarly to post-college-aged users with a high number of activity locations. In this case a linear model based on `activity_locations` and `age` will be confused - if an increase in `activity_days` leads to an increase in `p_open` when `age` is greater than 21, but leads to a decrease when `age` is less than 21, what should the sign on the coefficent of `activity_locations` be in the model?

If we created a new variable, $(\texttt{age} - 21) \cdot (\texttt{activity\_locations} - 8)$, it will be positive for highly-mobile post-college-aged users and static college-aged users. It will be negative for static post-college-aged users and highly-mobile college-aged users. If we already have the linear terms incorperated in our model, we can capture this behavior by simply adding the interaction term, without having to define a new variable.

```
f6 <- as.formula(p_open ~ activity_locations + age + age*activity_locations)
```

Compute MSE_{Train} and MSE_{Test} for this model. Create another model with several more interaction terms.

For each of our models we can look at our the models parameters by using `coef`. For example, we can inspect the first model's coefficiets using:

```
coef(fit.lm1)
```

What do you think the sign of the coefficent on the interaction term indicates?

Importance of randomness

To illustrate the importance of having a randomly selected training sample, let's try training on a subsample of the training data based on an age cutoff, ie, a non-representative sample of the population.

```
dd.train.nr <- dd.train[dd.train$age>26,]
y.train.nr <- dd.train.nr$p_open
```

To provide a fair comparison point for the non-random sample, we will create a random sample of the same size:

```
dd.train.r <- dd.train[sample(nrow(dd.train), nrow(dd.train.nr)),]
y.train.r <- dd.train.r$p_open
```

We will evaluate both the model trained on the non-random sample, and the model trained on the random sample against the same test data, namely '`dd.test`.

Begin, by using one of your models, e.g., `f2`, to compute MSE_{Train} and MSE_{Test} for the non-random training sample.

```
fit.lm2.nr <- lm(f2, dd.train.nr)

yhat.train.lm2.nr <- ...
mse.train.lm2.nr <- ...

yhat.test.lm2.nr <- ...
mse.test.lm2.nr <- ...
```

Now, repeat the same procedure using the random sample:

```
fit.lm2.r <- lm(f2, dd.train.nr)

yhat.train.lm2.r <- ...
mse.train.lm2.r <- ...

yhat.test.lm2.r <- ...
mse.test.lm2.r <- ...
```

How does `mse.test.lm2.nr` compare to `mse.test.lm2`? Can you explain what happened?

Continue exploring on your own

Try changing the seed and see which results stay the same, and which results change. If results are changing this indicates that the model may be overly sensitive.

Try thinking about ways you could systematicly decide what predictors you might choose to include in your model. What criteria would you use?