Introduction to Data Science

Homework 2

Student Name: Eric Niblock

Student Netid: ejn9259

Part 1: Case study (5 Points)

- Read this article (http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html) in the New York Times.
- Use what we've learned in class and from the book to describe how one could set Target's problem up as a predictive modeling problem, such that they could have gotten the results that they did. Formulate your solution as a proposed plan using our data science terminology. Include all the aspects of the data mining process, and be sure to include the motivation for predictive modeling and give a sketch of a solution. Be precise but concise.

------ANSWER-------Target's problem could be modeled as a predictive modeling problem by employing a supervised learning technique to answer the question: "Is this customer pregnant," where the outcome would be a binary 'yes' or 'no' with an associated probability. This outcome is the variable (label) we hope to predict, given various other variables (features) that we can collect from a consumer's visit to Target. The most important features would probably contain information about the purchase of certain products related to pregnancy, such as "scent-free soap and extra-big bags of cotton balls, in addition to hand sanitizers and washcloths." The goal is to train on a sample gathered from the population of pregnant customers, and not-pregnant customers, which can then be deployed on customers whose label we do not know (pregnant and not-pregnant).

The sample we hope to use can be taken from Target's baby shower registrar, with the consent of the residing mothers, to use as a group of individuals who have the label 'pregnant'. The sample related to those who are not-pregnant could reasonably be drawn from the whole population of Target consumers, unless they are on the baby shower registrar. Though there may be some individuals who are in fact pregnant within this sample, that number is likely to be low and negligible. All of the customers chosen will be assigned an ID, and purchasing history and future purchases can be attached to their account. Using these two groups, we can run some simple statistical analysis, such as getDfSummary() from below, in order to see if there is correlation between certain product purchases and being pregnant/not-pregnant. Then, these two samples can be combined, and split into training and validation sets. Using the results of our simple statistical analysis, we will have, hopefully, generated a rough idea of certain products which may signal that a mother is pregnant. We can then use these products as features, and attempt to train a model on our training data. The model can then be validated on the test data.

The motivation for creating such a model is obvious. Given the ability to predict if a certain consumer is pregnant, Target can target this consumer with specific advertising and promotions in order to increase sales with this population. Furthermore, as the article noted, after mothers have a child, shopping habits often change. So not only is there the benefit of increasing sales to pregnant mothers, but a successful predictive model of this sort has potential to draw in new shoppers for the rest of their lives.------ANSWER-------

Part 2: Exploring data in the command line (4 Points)

For this part we will be using the data file located in `"data/advertising_events.csv"`. This file consists of records that pertain to some online advertising events on a given day. There are 4 comma separated columns in this order: `userid`, `timestamp`, `domain`, and `action`. These fields are of type `int`, `int`, `string`, and `int` respectively. Answer the following questions using Linux/Unix bash commands. All questions can be answered in one line (sometimes, with pipes)! Some questions will have many possible solutions. Don't forget that in IPython notebooks you must prefix all bash commands with an exclamation point, i.e. `"!command arguments"`.

[Hints: You can experiment with whatever you want in the notebook and then delete things to construct your answer later. You can also use a bash shell (i.e., EC2 or a Mac terminal) and then just paste your answers here. Recall that once you enter the "!" then filename completion should work.]

[Here (https://opensource.com/article/17/2/command-line-tools-data-analysis-linux)](https://opensource.com/article/17/2/command-line-tools-data-analysis-linux) is a good linux command line reference.

1. How many records (lines) are in this file? (look up 'wc' command)

In [83]:
```
## Commands for this were run in Git Bash, and the code used was copied here with
!wc -l advertising_events.csv

## Output: 10341 advertising_events.csv
```

```
'wc' is not recognized as an internal or external command,
operable program or batch file.
```

2. How many unique users are in this file? (hint: consider the 'cut' command and use pipe operator '|')

In [84]:
```
## Commands for this were run in Git Bash, and the code used was copied here with
!cut -d ',' -f 1 advertising_events.csv | uniq -u | wc -l

## Output: 10321
```

```
'cut' is not recognized as an internal or external command,
operable program or batch file.
```

3. Rank all domains by the number of visits they received in descending order. (hint: consider the 'cut', 'uniq' and 'sort' commands and the pipe operator).

```
In [ ]: ## Commands for this were run in Git Bash, and the code used was copied here with
        !cut -d ',' -f 3 advertising_events.csv | sort | uniq -c | sort -r

        ## Output:
        # 3114 google.com
        # 2092 facebook.com
        # 1036 youtube.com
        # 1034 yahoo.com
        # 1022 baidu.com
        # 513 wikipedia.org
        # 511 amazon.com
        # 382 qq.com
        # 321 twitter.com
        # 316 taobao.com
```

4. List all records for the user with user id 37. (hint: this can be done using 'grep')

```
In [ ]: ## Commands for this were run in Git Bash, and the code used was copied here with
        !awk -F, '{if ($1 == 37) print $0}' advertising_events.csv

        ## Output:
        # 37,648061658,google.com,0
        # 37,642479972,google.com,2
        # 37,644493341,facebook.com,2
        # 37,654941318,facebook.com,1
        # 37,649979874,baidu.com,1
        # 37,653061949,yahoo.com,1
        # 37,655020469,google.com,3
        # 37,640878012,amazon.com,0
        # 37,659864136,youtube.com,1
        # 37,640361378,yahoo.com,1
        # 37,653862134,facebook.com,0
        # 37,648828970,youtube.com,0
```

Part 3: Dealing with data Pythonically (16 Points)

1. (1 Point) Download the data set `"data/ads_dataset.tsv"` and load it into a Python Pandas data frame called `ads`.

```
In [1]: import pandas as pd

        ads = pd.read_csv( r'C:\Users\Eric\ads_dataset.tsv', sep='\t')
```

2. (4 Points) Write a Python function called `getDfSummary()` that does the following:

- Takes as input a data frame
- For each variable in the data frame calculates the following features:
  - `number_nan` to count the number of missing not-a-number values
  - Ignoring missing, NA, and Null values:
    - `number_distinct` to count the number of distinct values a variable can take on

    - `mean`, `max`, `min`, `std` (standard deviation), and `25%`, `50%`, `75%` to correspond to the appropriate percentiles
- All of these new features should be loaded in a new data frame. Each row of the data frame should be a variable from the input data frame, and the columns should be the new summary features.
- Returns this new data frame containing all of the summary information

Hint: The pandas `describe()` method returns a useful series of values that can be used here.

In [2]:
```
def getDfSummary(input_data):
    nans = []
    for c in input_data.columns:
        nans.append(input_data[c].isna().sum())
    df = pd.DataFrame({'number_nan':nans})
    df = df.transpose()
    df.columns = input_data.columns

    s = input_data.describe()
    s = s.append(df)

    unique = []
    for c in input_data.columns:
        unique.append(input_data[c].nunique())
    df = pd.DataFrame({'number_distinct':unique})
    df = df.transpose()
    df.columns = input_data.columns
    s = s.append(df)
    s = s.transpose()
    s = s.drop(labels='count', axis=1)

    return(s)
```

3. How long does it take for your `getDfSummary()` function to work on your `ads` data frame? Show us the results below.

Hint: `use %timeit`

In [3]:
```
%timeit getDfSummary(ads)
```

```
92.2 ms ± 2.33 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

4. (2 Points) Using the results returned from `getDfSummary()`, which fields, if any, contain missing `NaN` values?

In [14]:
```
n = getDfSummary(ads)
n[n['number_nan'] > 0]

## This code shows that 'buy_freq' is the only field which contains nan values
```

Out[14]:

	mean	std	min	25%	50%	75%	max	number_nan	number_distinct
buy_freq	1.240653	0.782228	1.0	1.0	1.0	1.0	15.0	52257.0	10.0

5. (4 Points) For the fields with missing values, does it look like the data is missing at random? Are there any other fields that correlate perfectly, or make it more likely that the data is missing? If missing, what should the data value be? Don't just show code here. Please explain your answer. [Edit this to ask for more details on why they are 0]

Hint: create another data frame that has just the records with a missing value. Get a summary of this data frame using `getDfSummary()` and compare the differences. Do some feature distributions change dramatically?

In [15]:
```
getDfSummary(ads)
```

Out[15]:

	mean	std	min	25%	50%	75%	max	nu
isbuyer	0.042632	0.202027	0.0000	0.0	0.0	0.000000	1.00000	
buy_freq	1.240653	0.782228	1.0000	1.0	1.0	1.000000	15.00000	
visit_freq	1.852777	2.921820	0.0000	1.0	1.0	2.000000	84.00000	
buy_interval	0.210008	3.922016	0.0000	0.0	0.0	0.000000	174.62500	
sv_interval	5.825610	17.595442	0.0000	0.0	0.0	0.104167	184.91670	
expected_time_buy	-0.198040	4.997792	-181.9238	0.0	0.0	0.000000	84.28571	
expected_time_visit	-10.210786	31.879722	-187.6156	0.0	0.0	0.000000	91.40192	
last_buy	64.729335	53.476658	0.0000	18.0	51.0	105.000000	188.00000	
last_visit	64.729335	53.476658	0.0000	18.0	51.0	105.000000	188.00000	
multiple_buy	0.006357	0.079479	0.0000	0.0	0.0	0.000000	1.00000	
multiple_visit	0.277444	0.447742	0.0000	0.0	0.0	1.000000	1.00000	
uniq_urls	86.569343	61.969765	-1.0000	30.0	75.0	155.000000	206.00000	
num_checkins	720.657592	1275.727306	1.0000	127.0	319.0	802.000000	37091.00000	
y_buy	0.004635	0.067924	0.0000	0.0	0.0	0.000000	1.00000	

In [16]:
```
ads_nan = ads[ads['buy_freq'].isnull()]
getDfSummary(ads_nan)
```

Out[16]:

	mean	std	min	25%	50%	75%	max	nu
isbuyer	0.000000	0.000000	0.0000	0.0	0.0	0.000000	0.00000	
buy_freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
visit_freq	1.651549	2.147955	1.0000	1.0	1.0	2.000000	84.00000	
buy_interval	0.000000	0.000000	0.0000	0.0	0.0	0.000000	0.00000	
sv_interval	5.686388	17.623555	0.0000	0.0	0.0	0.041667	184.91670	
expected_time_buy	0.000000	0.000000	0.0000	0.0	0.0	0.000000	0.00000	
expected_time_visit	-9.669298	31.239030	-187.6156	0.0	0.0	0.000000	91.40192	
last_buy	65.741317	53.484622	0.0000	19.0	52.0	106.000000	188.00000	
last_visit	65.741317	53.484622	0.0000	19.0	52.0	106.000000	188.00000	
multiple_buy	0.000000	0.000000	0.0000	0.0	0.0	0.000000	0.00000	
multiple_visit	0.255602	0.436203	0.0000	0.0	0.0	1.000000	1.00000	
uniq_urls	86.656180	61.996711	-1.0000	30.0	75.0	155.000000	206.00000	
num_checkins	721.848518	1284.504018	1.0000	126.0	318.0	803.000000	37091.00000	
y_buy	0.003024	0.054904	0.0000	0.0	0.0	0.000000	1.00000	

------ANSWER------- Many features change when we observe a dataframe with only the rows containing nan values. Observe the variables 'isbuyer', 'buy_interval', 'expected_time_buy', and 'multiple_buy', as they correlate perfectly with 'buy_freq' being a nan value - specifically, these values are all zero when 'buy_freq' is a nan value. All of the nan values should be replaced with zeros, since 'buy_freq' being nan seems to suggest that that an individual has not made a purchase (since all of the correlated variables would, in fact, be zero, if 'buy_freq' was zero), and thus zero would represent this. ------ANSWER-------

6. (4 Points) Which variables are binary?

In [82]:
```
c = getDfSummary(ads)
c[c['number_distinct'] == 2]

## This code shows that 'isbuyer', 'multiple_buy', 'multiple_visit', and 'y_buy'
```

Out[82]:

	mean	std	min	25%	50%	75%	max	number_nan	number_distinct
isbuyer	0.042632	0.202027	0.0	0.0	0.0	0.0	1.0	0.0	2.0
multiple_buy	0.006357	0.079479	0.0	0.0	0.0	0.0	1.0	0.0	2.0
multiple_visit	0.277444	0.447742	0.0	0.0	0.0	1.0	1.0	0.0	2.0
y_buy	0.004635	0.067924	0.0	0.0	0.0	0.0	1.0	0.0	2.0