Comparison on Online Advertising Channels: Data Analysis on Avazu Mobile Ads Data

Linna Li

Date: 2016/03/18
Word count: 2244

Introduction

Online advertising, or Computational advertising, is a form of marketing and advertising that applies the website or other internet access to deliver promotional information to consumers. Since all the activities happen through the internet and devices, data collection and analytics analysis can be performed for targeting optimization, which is one of the advantages of Online advertising versus traditional methods. Another competitive advantage is that with the use increase on different mobile devices, more online advertising channels emerges in recent years and greatly expand the exposure scale of advertisements.

The objective of this research is following: a. explore possible methods to discover knowledge from a mobile advertising dataset; b. compare different mobile ads channels; c. determine the important predictors for click; d. draw conclusions based on the research. In this research, the mobile ads dataset from Avazu will be analyzed with various data analysis methods for online advertising channel comparison. And the conclusion will also cover the special features of advertising data.

Dataset

Original Data

The dataset used in this research is from the Kaggle competition *Click-Through Rate Prediction* [1] starting on November 2014. It describes the 10-day click-through data of mobile website and application advertising which is ordered chronologically. For the competition purpose, the non-clicks and clicks are subsampled according to different strategies by the data provider. The original set contains 40,428,967 records and 24 attributes and the file is as large as 6GB which is too much for my computer. So before conducting analysis on it, I will prepare it first for my research purpose. Here are the column information:

Column Group	Column Names
Click	Click (0 - 1)
Time related	HOUR
Channels	site_id, site_domain, site_category, app_id, app_domain, app_category
Device	device_id, device_ip, device_model, device_type, device_connect_type
Anonymized categorical variables	C1, C14, C15, C16, C17, C18, C19, C20, C21

Data Preparation

In this part, I will describe the data preparation techniques that are applied on the original dataset. All the data preparation works are finished with Python (Jupyter).

Data Balance and Data Reduction

Typically, the online advertising data is very unbalanced because it's difficult to attract the right customers with the right ads. This phenomenon happens in this data too. There are 33,563,901 non-clicked and 6,865,066 clicked data. To ensure a fast analysis speed, I determined the total number of cases as 300,000. And to make the target balanced, I randomly sampled 150,000 records from the non-click subset and 150,000 records form the click subset. Therefore, the reduced data is nearly 0.8% of the original data.

Missing Values

The missing values in this dataset is represented with some code, such as '85f751fd' and '7801e8d9', and the codes are changing for different columns. Thus, I detected those missing value codes and then replaced them with 0.

Feature Changes

The column HOUR is in the format of YYMMDDHH, which provides the information about when a particular ads was displayed. The date covers 10 days from 2014/10/21 to 2014/10/30 with 24 hours data. Thus, I derived two more columns based on this, DayOfMonth and HourOfDay.

Data Split

After the preprocessing, the final data contains 300,000 records with 24 attributes. In addition, as the main purpose of the research is to compare different channels of website and application, I also tried to extract the web-click and app-click information from the overall data. Based on the previous preparation, the site-related and app-related columns are like this:

		Rows									
		299991	299992	299993	299994	299995	299996	299997	299998	299999	300000
Columns	site	0	d9750ee7	0	0	8ab5bdcf	0	0	0	0	1fbe01fe
		0	98572c79	0	0	db11867b	0	0	0	0	f3845767
		0	f028772b	0	0	c0dd3be3	0	0	0	0	28905ebd
	app	6efb59d5	0	e2fcccd2	53de0284	0	f888bf4c	f0d41ff1	9c13b419	e2fcccd2	0
		d9b5648e	0	5c5a694b	d9b5648e	0	5b9c592b	2347f47a	2347f47a	5c5a694b	0
		0f2161f8	0	0f2161f8	0f2161f8	0	0f2161f8	0f2161f8	f95efa07	0f2161f8	0

As you can see from the columns, the records in the data are either about web-click or about app-click. This is the base ground of this research and data extraction. As a result, the web-data contains 204,272 records and the app-data contains 95,729 records.

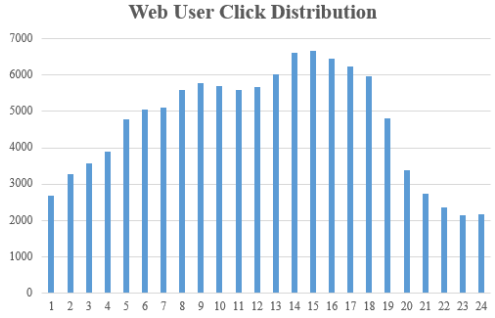
Thus, with the data preprocessing, I have three data sets to work on: overall-data, web-data and app-data. For the future use, I also randomly split them into training and test sets with the ratio of 80/20.

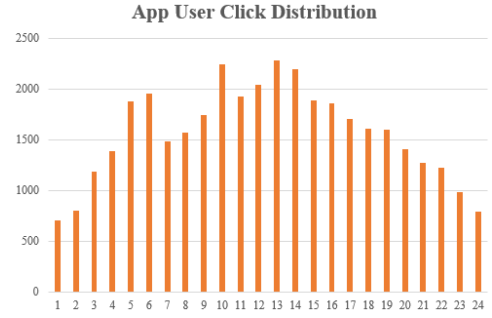
Experiments

In this section, I will perform analysis on the datasets with different techniques to compare web-data and app-data, including descriptive analysis, Click-Through Rate analysis, click prediction and social network analysis. The first two parts are conducted with Python in Jupyter and the other two parts are in R and Gephi separately.

Click Distribution of Channels

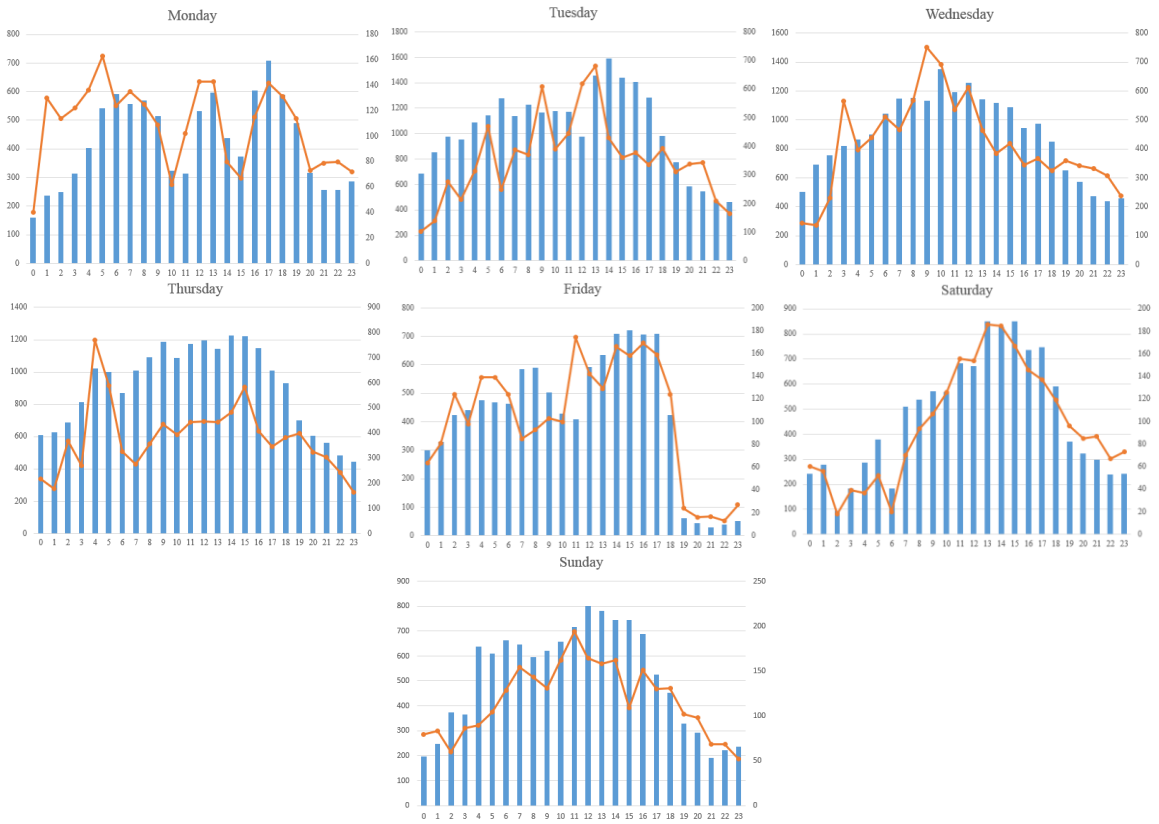
With the derived time-related column HourOfDay, I want to find out the user behavior during a general day by checking the hourly click number. Thus, for both web-data and app-data, I grouped them with hour number, from 1 to 24, and aggregated the click based on that. The plots below show the results:





As we can see from the click distributions by hour, the web and app users behave in different ways. First, the highest number of clicks are great different. With a randomly sampled data, this indicates that the majority mobile ads clicks occurred via websites. Secondly, the peak of web-data happens around 2PM-3PM, while the peak of app-data appears in the early morning, like 5AM/6AM, and at the lunch time. Third, the number of web-click dramatically decreases after work.

Then, I further grouped the click data with the column DayOfWeek to see if the user behavior will change during a week. In the charts below, the blue bars represent the web-data and the orange lines are app-data.



These seven plots reflect the web-click and app-click distributions by hour from Monday to Sunday. The left Y-axis is for bars and the right is for lines. Basically, the trends of the two channels are similar to each other. But when we look them closer, there are some interesting findings. From the small total click number of Monday, we can see that people are not likely to view ads during the early morning. But no matter what day it is, the click roars from the noon hours to 4PM, and the app-click peak occur a little earlier than the web-click. On Thursday, there is an obvious increase on app-click around 4AM. And on Friday after work and Saturday morning, it seems that people are too busy to click on ads.

Simulated Click-Through Rate

Click-Through Rate is an important metric for online advertising. Its definition is that the percentage of people visiting a web page who access a hypertext link to a particular advertisement. The CTR typically equals to the division of the number of click on a particular ads by the total number of impression of this ads.

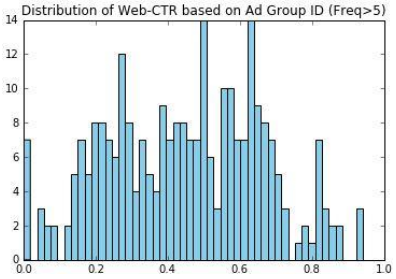
However, the data used here doesn't have the impression information. So I will simulate the calculation of CTR based on the following points:

a. Simulated CTR = (Aggregated number of Click-1) / (Total number of appearance freq in dataset)
b. With external information, C14 contains Ad ID and C17 contains Ad Group ID. Different from the traditional CTR, the Ad Group will also be the click target.
c. Since the original data is sampled by the data provider first, there are much more click data than in reality. Thus, the values of CTR must higher than the typical baseline.
d. Threshold will be set for data to avoid lots of 100% CTR. The Ad Group IDs must appear more than 5 times and the Ad IDs much appear more than 15 times in the data. Otherwise, the IDs will be filtered out.

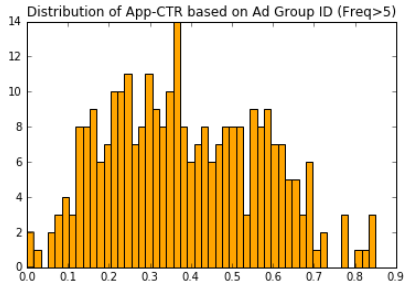
In the following, the results of CTR for Ad Group and CTR for Ad will be shown.

Based on Ad Group ID (C17)

The total numbers of unique Ad Group ID for web-data and app-data are 297 and 302 respectively. For web-data, the CTRs mostly locate within 3 groups: (0.2, 0.3), (0.4, 0.5) and (0.55, 0.65). But compared to app-data, there are more high CTRs in web-data. We can also tell this from the highest/lowest CTR values with the top 10 ad groups. Also, the greyed items are the ad groups with high CTRs for both channels.



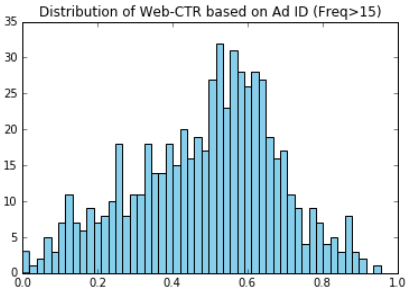
web_Group	Total	Click	CTR
1694	101	95	0.940594
2518	45	42	0.933333
2662	26	24	0.923077
2286	1008	886	0.923077
2295	2850	2474	0.86807
827	262	225	0.858779
2101	7	6	0.858779
2569	400	338	0.845
1903	24	20	0.833333
2659	6	5	0.833333



app_Group	Total	Click	CTR
2659	20	17	0.85
1926	126	106	0.84127
2295	119	100	0.840336
2702	11	9	0.818182
2438	10	8	0.8
2638	18	14	0.777778
2421	31	24	0.774194
2331	190	146	0.768421
1272	152	109	0.717105
2510	7	5	0.714286

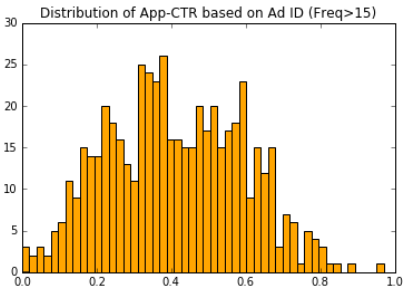
Based on Ad ID (C14)

The total numbers of unique Ad ID for web-data and app-data are 1416 and 1483 respectively. When looking at the distributions, we will see that both of them are more concentrated. For web-data, most of CRTs are around 0.5 to 0.7 which could be a high score. And for app-data, the CRTs are mostly around 0.3 to 0.4. When comparing the highest/lowest CTR in the top20 table, we can see the CTR of app-data decrease very fast. In addition, there are two shared Ads. Since the amount of ads is large, they could be really attractive to users no matter under what situation.



	web_ID	Total	Click	CTR
1	21814	22	21	0.9545
2	21812	23	21	0.9130
3	17759	20	18	0.9000
4	20018	503	446	0.8867
5	22272	171	151	0.8830
6	20635	305	269	0.8820
7	21277	301	264	0.8771
8	21914	16	14	0.8750
9	8996	80	70	0.8750
10	22270	70	61	0.8714

	web_ID	Total	Click	CTR
11	20019	505	440	0.8713
12	20093	2850	2474	0.8681
13	23087	22	19	0.8636
14	21918	22	19	0.8636
15	8994	85	73	0.8588
16	23376	33	28	0.8485
17	8995	97	82	0.8454
18	22804	24	20	0.8333
19	20015	64	53	0.8281
20	20346	1410	1166	0.8270



	app_ID	Total	Click	CTR
1	16989	33	32	0.9697
2	18936	27	24	0.8889
3	20093	119	100	0.8403
4	20274	29	24	0.8276
5	21590	16	13	0.8125
6	20215	26	21	0.8077
7	21791	254	205	0.8071
8	18925	29	23	0.7931
9	20009	37	29	0.7838
10	22756	18	14	0.7778

	app_ID	Total	Click	CTR
11	23553	18	14	0.7778
12	12472	31	24	0.7742
13	20345	100	77	0.7700
14	21677	39	30	0.7692
15	20346	90	69	0.7667
16	23216	37	28	0.7568
17	21595	20	15	0.7500
18	20271	38	28	0.7368
19	18934	37	27	0.7297
20	21273	92	67	0.7283

Click Prediction Analysis

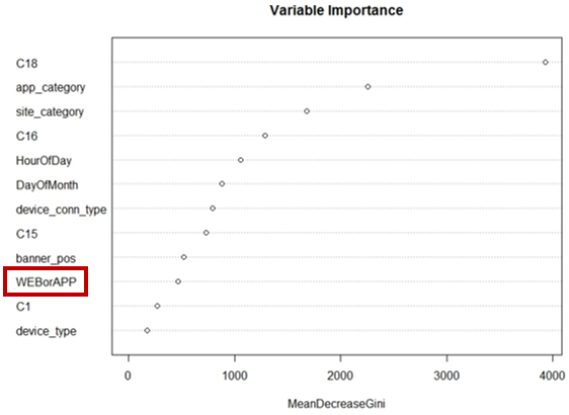
By performing prediction analysis on click, I aim to determine the important predictors for both web click data and app click data. To have to full picture on the issue, the analysis will be done on three data sets: the overall-data, web-data and app-data. Decision Tree and Random Forest will be applied for comparison.

Prediction on Overall-data

In the overall data, I added a new binary attribute into the dataset, WEBorAPP, which is used to define if a case is about web click or app click. If the case is about web, its values in the new attribute is "1"; otherwise, it would be "0". With WEBorAPP, I want to see if the channel is an important factor for clicks. Also, due to the computation power of R, the categorical attributes with more than 32 levels are taken down.

From the results below, the performance of classification is improved a lot with Random Forest. The plot of variable importance is from Random Forest in R. We can find that the new variable, WEBorAPP is indeed an important factor for click among all the 25 variables.

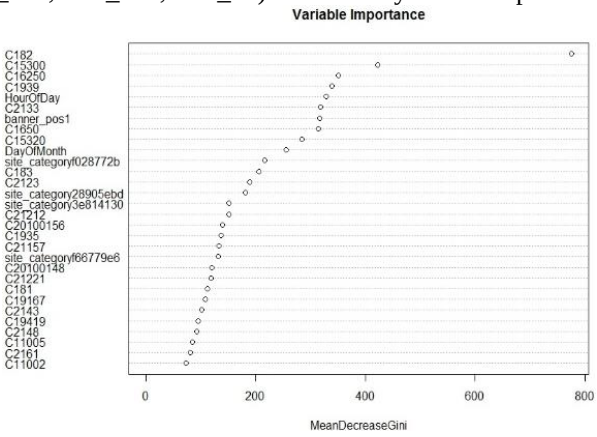
overall	Decision Tree	Random Forest (ntree=300)
Accuracy	0.6098	0.6509
Sensitivity	0.6098	0.6509
Specificity	0.6081	0.6591



Prediction on Web-data

The web-data is the underlying dataset in this part. I took down all the categorical attributes with more than 50 levels and convert the data into dummy data. Basically, the metric scores are close to the overall data prediction and RF still outperforms DT. As to variable importance, it's interesting that the top4 factors are all about the nature of advertisement (C18_2, C15_300, C16_250, C19_39). HourOfDay is also important.

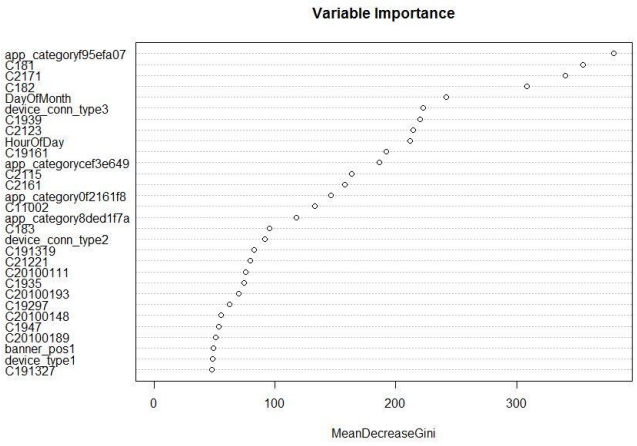
web	Decision Tree	Random Forest (ntree=150)
Accuracy	0.6105	0.6614
Sensitivity	0.5932	0.6650
Specificity	0.6233	0.6546



Prediction on App-data

The app-data is the underlying dataset in this part. Except for the C-X attributes, the Application Category, DayOfMonth, Device Connection Type and HourOfDay are also among the top10 factors. This is a little different with the result of web-data. As to the models, Random Forest improves the performance a lot, especially on the Specificity.

app	Decision Tree	Random Forest (ntree=150)
Accuracy	0.6528	0.6825
Sensitivity	0.6764	0.6110
Specificity	0.5889	0.7211



Social Network Analysis

Since every record in the dataset contains a "source" ID and an ads ID, it would be interesting to apply Social Network analysis here with Gephi. To plot a meaningful network, I first filtered the data with some criteria. First, I will selected the pairs from "clicked" data. Second, the appearance of both Source ID and Target Ad ID should be more than 15 to avoid generating huge networks.

The resulting network plots are in the Appendix. Even the screenshot is not very clear, we still can tell the connection patterns of the two channels are different. For example, the network of app-data has more isolated groups than the web-data, indicating an accurate targeting is more important for app-ads.

Conclusion

In this research, I performed different data analysis on the Avazu mobile advertising dataset to compare website click and application click data from various perspectives. As a result, I found that more clicks occur via mobile websites, especially during the daytime. And people tend to click ads by apps in the early morning and evening. As to the important predictors, good timing is one of the top factor for both web-data and app-data. But there is also difference between this two channels. For web-data, the characteristics of ads itself are important for predicting click or non-click. For app-data, some external influencers are crucial, such as app category and device connection type. Regarding to the prediction methods, Random Forest can provide evident improvement for the results based on Decision Tree. And Social Network can be a good choice for click data pattern discovery.

In data analysis process, I met some difficulties with the data set. First of all, the original dataset is too large to use within normal machine or data tools like R. My solution is sampling on the data. But lots of information is lost at this stage, which directly leads to a poor accuracy of models. Then, the data is very unbalanced, which will lower the Specificity in prediction. Third, it's special for online data that IP and ID information matters, so we have to deal with extremely-high level attributes. Also, for this particular Kaggle data, many anonymized attributes weaken the models' interpretation ability and make many solutions unusable.

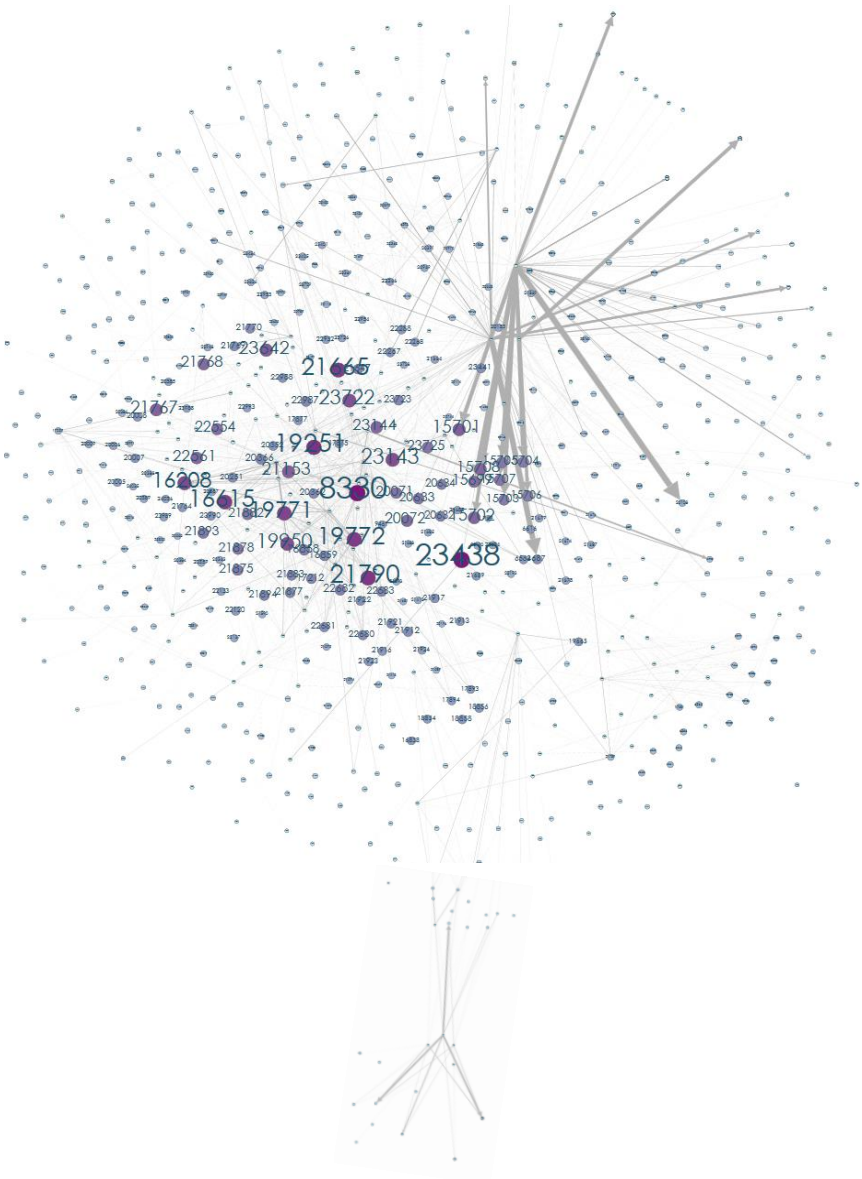
In the future work, I will try to include more data to improve the current results, and more predict algorithms will be tested on them too. To deal with unbalanced data, penalty parameters can be added into the classification models. Additionally, the comparison on channels can be conducted from more aspects.

Reference

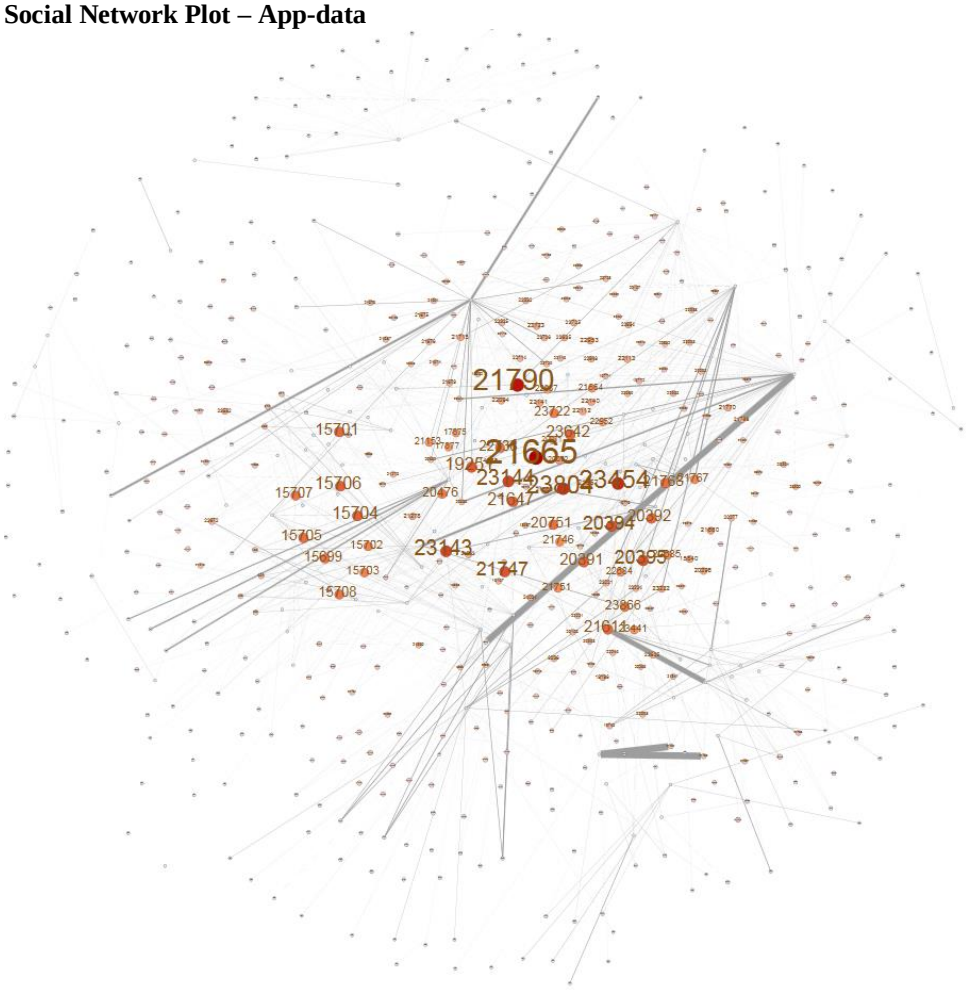
[1] Kaggle Competition: Click-Through Rate Prediction. (2014), https://www.kaggle.com/c/avazu-ctr-prediction/data
[2] Mobile Marketer (2012), *Classic Guide to Mobile Advertising*
[3] Vallina-Rodriguez, N. & Shah, J. & Finamore, A. & Grunenberger, Y. & Haddadi, H. & Papagiannaki, K. & Crowcroft, J. (2012), Breaking For Commercials: Characterizing Mobile Advertising
[4] Powers, D. & Xie, Y. (1999), Statistical Methods for Categorical Data Analysis
[5] Berrar, D. (2012), Random forests for the detection of click fraud in online mobile advertising

Appendix

Social Network Plot – Web-data



Social Network Plot – App-data



Python Code I – Data Preparation

1. Extract the click and non-click data sets.

```
cd "G:\Dear data\ClickThrough\data"
```

```
f = open('train.csv')
c = open('click.csv', 'wb' )
n = open('nonclick.csv', 'wb' )
```

```
r = csv.reader(f)
wc = csv.writer(c, delimiter=',')
wn = csv.writer(n, delimiter=',')
```

```
for line in r:
    if line[1] == '1':
        wc.writerow(line)
    else:
        wn.writerow(line)
```

```
f.close()
c.close()
n.close()
```

2. Sampling 150,000 records form Click and Non-click data separately.

```
import pandas
import random
```

```
n = 6865066 #number of records in file
s = 150000 #desired sample size
filename = "click.csv"
skip = sorted(random.sample(xrange(n),n-s))
ran_click = pandas.read_csv(filename, skiprows=skip,
                            names = ['id','click','hour','C1','banner_pos','site_id','site_domain','site_category',
                                     'app_id','app_domain','app_category','device_id','device_ip','device_model',
                                     'device_type','device_conn_type','C14','C15','C16','C17','C18','C19','C20','C21'])
```

```
ran_click.shape
```

```
(150000, 24)
```

```
n = 33563901 #number of records in file
s = 150000 #desired sample size
filename = "nonclick.csv"
skip = sorted(random.sample(xrange(n),n-s))
ran_nonclick = pandas.read_csv(filename, skiprows=skip,
                            names = ['id','click','hour','C1','banner_pos','site_id','site_domain','site_category',
                                     'app_id','app_domain','app_category','device_id','device_ip','device_model',
                                     'device_type','device_conn_type','C14','C15','C16','C17','C18','C19','C20','C21'])
```

```
ran_nonclick.shape
```

```
(150001, 24)
```

3. Combine the Click and Non-click data into a new dataset.

```
newdata_ran = pd.concat([ran_click, ran_nonclick], axis=0)
```

```
newdata_ran.head(2)
```

	id	click	hour	C1	banner_pos	site_id	site_domain	site_category	app_id	app_domain	...
0	10028335185239032448	1	14102100	1005	1	d9750ee7	98572c79	f028772b	ecad2386	7801e8d9	...
1	10046495301432444321	1	14102100	1005	1	7dd19f44	9690165f	f028772b	ecad2386	7801e8d9	...

2 rows × 24 columns

```
newdata_ran['click'].value_counts()
```

```
0    150001
1    150000
Name: click, dtype: int64
```

```
newdata_ran.to_csv("randomnewdata.csv", columns=header, index=False)
```

4. Replace the missing values.

```
siteID = mydatanew['site_id'].replace('85f751fd', 0).replace('84c7ba46', 'other') # 2 types of missing values
siteDomain = mydatanew['site_domain'].replace('c4e18dd6', 0)
siteCategory = mydatanew['site_category'].replace('50e219e0', 0)

appID = mydatanew['app_id'].replace('ecad2386', 0)
appDomain = mydatanew['app_domain'].replace('7801e8d9', 0)
appCategory = mydatanew['app_category'].replace('07d7df22', 0)
```

5. Generate derived columns of DayOfWeek and HourOfDay.

```
dayofmonth = mydata['hour'].astype('str').str[4:6]
```

```
dayofmonth.value_counts()
```

```
22    38585
28    37517
30    31348
21    30825
23    29768
26    29124
29    27366
25    25955
24    24826
27    24687
Name: hour, dtype: int64
```

```
hourofday = mydata['hour'].astype('str').str[6:8]
```

```
hourofday.value_counts()
```

```
13    17678
14    16580
09    16577
12    16529
15    16008
08    15566
10    15488
16    15483
11    15283
17    14947
05    14416
04    13949
07    13909
06    13164
18    12826
03    10357
19     9675
02     9163
20     8175
01     7728
21     7360
00     6576
22     6517
23     6047
Name: hour, dtype: int64
```

```
mydata.insert(3, 'DayOfMonth', dayofmonth)
mydata.insert(4, 'HourOfDay', hourofday)
```

```
mydatanew = mydata.drop(['hour'], 1)
```

6. Create the new binary attribute.

```
siteID = mydatanew['site_id'].replace('85f751fd', 0).replace('84c7ba46', 'other') # 2 types of missing values
webORapp = siteID
```

```
webORapp.dtype
```

```
dtype('O')
```

```
webORapp[webORapp != 0] = 1
```

```
webORapp.value_counts()
```

```
1    204272
0     95729
Name: site_id, dtype: int64
```

7. Split the data into Web-data and App-data.

```
mydatanew.groupby(['WEBorAPP'])['click'].sum()
```

```
WEBorAPP
0     37831
1    112169
Name: click, dtype: int64
```

```
webdata = mydatanew[mydatanew['WEBorAPP']==1]
appdata = mydatanew[mydatanew['WEBorAPP']==0]
```

Python Code II – Click Distribution

1. DayOfWeek

```
webmonday = webdata[(webdata['DayOfMonth'] == 27)]
webtuesday = webdata[(webdata['DayOfMonth'] == 21) | (webdata['DayOfMonth'] == 28)]
webwednesday = webdata[(webdata['DayOfMonth'] == 22) | (webdata['DayOfMonth'] == 29)]
webthursday = webdata[(webdata['DayOfMonth'] == 23) | (webdata['DayOfMonth'] == 30)]
webfriday = webdata[(webdata['DayOfMonth'] == 24)]
websaturday = webdata[(webdata['DayOfMonth'] == 25)]
websunday = webdata[(webdata['DayOfMonth'] == 26)]
```

```
appmonday = appdata[(appdata['DayOfMonth'] == 27)]
apptuesday = appdata[(appdata['DayOfMonth'] == 21) | (appdata['DayOfMonth'] == 28)]
appwednesday = appdata[(appdata['DayOfMonth'] == 22) | (appdata['DayOfMonth'] == 29)]
appthursday = appdata[(appdata['DayOfMonth'] == 23) | (appdata['DayOfMonth'] == 30)]
appfriday = appdata[(appdata['DayOfMonth'] == 24)]
appsaturday = appdata[(appdata['DayOfMonth'] == 25)]
appsunday = appdata[(appdata['DayOfMonth'] == 26)]
```

2. HourOfDay

```
appsunday.groupby(['HourOfDay'])['click'].sum()
```

```
HourOfDay
0      79
1      83
2      60
3      86
4      89
5     104
6     128
7     154
8     143
9     131
10    162
11    194
12    164
13    158
14    162
15    109
16    151
17    130
18    131
19    102
20     98
21     68
22     68
23     52
Name: click, dtype: int64
```

Python Code III – Simulated Click-Through Rate

1. Based on Ad Group ID

```
len(mydata['C17'].unique()), len(webdata['C17'].unique()), len(appdata['C17'].unique())
```

```
(409, 310, 332)
```

```
webtotal = webdata.groupby(['C17'])['click'].count()
webclicked = webdata.groupby(['C17'])['click'].sum()
webgroupid = list(sorted(webdata['C17'].unique()))

apptotal = appdata.groupby(['C17'])['click'].count()
appclicked = appdata.groupby(['C17'])['click'].sum()
appgroupid = list(sorted(appdata['C17'].unique()))
```

```
webtotalnum = list(webtotal)
webclicknum = list(webclicked)

apptotalnum = list(apptotal)
appclicknum = list(appclicked)
```

```
webframe = pd.DataFrame({"webGroup":webgroupid, "Total":webtotalnum, "Click":webclicknum}, columns=['webGroup','Total','Click'])
appframe = pd.DataFrame({"appGroup":appgroupid, "Total":apptotalnum, "Click":appclicknum}, columns=['appGroup','Total','Click'])
```

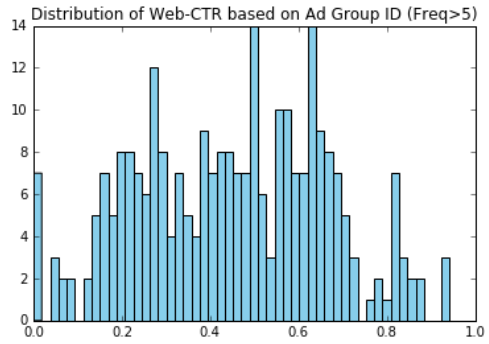
```
webctr_adgroup_data = webframe[webframe['Total'] > 5]
appctr_adgroup_data = appframe[appframe['Total'] > 5]
```

```
webctr_adgroup = webctr_adgroup_data['Click']/webctr_adgroup_data['Total']
webctr_adgroup_data['CTR'] = webctr_adgroup

appctr_adgroup = appctr_adgroup_data['Click']/appctr_adgroup_data['Total']
appctr_adgroup_data['CTR'] = appctr_adgroup
```

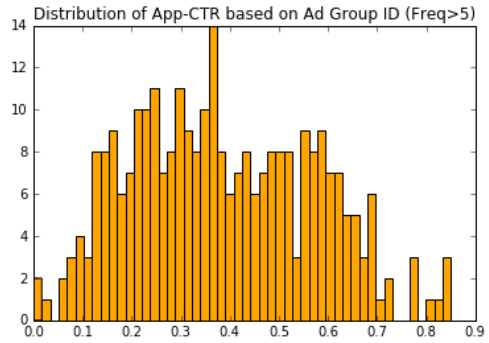
```
plt.hist(webctr_adgroup_data['CTR'], bins=50, color="skyblue")
plt.title('Distribution of Web-CTR based on Ad Group ID (Freq>5)')
```

```
<matplotlib.text.Text at 0x1419a668>
```




```
plt.hist(appctr_adgroup_data['CTR'], bins=50, color="orange")
plt.title('Distribution of App-CTR based on Ad Group ID (Freq>5)')
```

```
<matplotlib.text.Text at 0x12395c18>
```



2. Top Ranking

```
webctr_group_rank = webctr_adgroup_data.sort(['CTR'], ascending=[0])
appctr_group_rank = appctr_adgroup_data.sort(['CTR'], ascending=[0])
```

```
C:\Users\Nicole\Anaconda2\lib\site-packages\ipykernel\__main__.py:1:
values(by=.....)
  if __name__ == '__main__':
C:\Users\Nicole\Anaconda2\lib\site-packages\ipykernel\__main__.py:2:
values(by=.....)
  from ipykernel import kernelapp as app
```

```
webctr_group_rank.head(10)
```

	webGroup	Total	Click	CTR
33	1694	101	95	0.940594
164	2518	45	42	0.933333
254	2662	26	24	0.923077
99	2286	1008	886	0.878968
103	2295	2850	2474	0.868070
13	827	262	225	0.858779
71	2101	7	6	0.857143
198	2569	400	338	0.845000
51	1903	24	20	0.833333
251	2659	6	5	0.833333

```
appctr_group_rank.head(10)
```

	appGroup	Total	Click	CTR
267	2659	20	17	0.850000
55	1926	126	106	0.841270
96	2295	119	100	0.840336
294	2702	11	9	0.818182
128	2438	10	8	0.800000
247	2638	18	14	0.777778
119	2421	31	24	0.774194
106	2331	190	146	0.768421
28	1272	152	109	0.717105
169	2510	7	5	0.714286

3. Based on Ad ID

```
len(mydata['C14'].unique()), len(webdata['C14'].unique()), len(appdata['C14'].unique())
```

```
(1965, 1416, 1483)
```

```
webtotal2 = webdata.groupby(['C14'])['click'].count()
webclicked2 = webdata.groupby(['C14'])['click'].sum()
webadid = list(sorted(webdata['C14'].unique()))

apptotal2 = appdata.groupby(['C14'])['click'].count()
appclicked2 = appdata.groupby(['C14'])['click'].sum()
appadid = list(sorted(appdata['C14'].unique()))
```

```
webtotalnum2 = list(webtotal2)
webclicknum2 = list(webclicked2)

apptotalnum2 = list(apptotal2)
appclicknum2 = list(appclicked2)
```

```
webframe2 = pd.DataFrame({"webAds":webadid, "Total":webtotalnum2, "Click":webclicknum2}, columns=['webAds','Total','Click'])
appframe2 = pd.DataFrame({"appAds":appadid, "Total":apptotalnum2, "Click":appclicknum2}, columns=['appAds','Total','Click'])
```

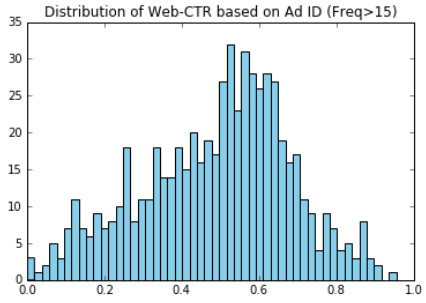
```
webctr_adid_data = webframe2[webframe2['Total'] > 15]
appctr_adid_data = appframe2[appframe2['Total'] > 15]
```

```
webctr_adid = webctr_adid_data['Click']/webctr_adid_data['Total']
webctr_adid_data['CTR'] = webctr_adid

appctr_adid = appctr_adid_data['Click']/appctr_adid_data['Total']
appctr_adid_data['CTR'] = appctr_adid
```

```
plt.hist(webctr_adid_data['CTR'], bins=50, color="skyblue")
plt.title('Distribution of Web-CTR based on Ad ID (Freq>15)')
```

```
<matplotlib.text.Text at 0x218f87f0>
```




```
plt.hist(appctr_adid_data['CTR'], bins=50, color="orange")
plt.title('Distribution of App-CTR based on Ad ID (Freq>15)')
```

```
<matplotlib.text.Text at 0x21bcfd30>
```