Report

January 15, 2018

1 Travel Audience

Data Science Recruitment Challenge

Synopsis The main purpose of this report is to understand who clicks on an ad. A set of features which might be used in predictive modeling will be computed. Rules and logic behind each feature will be briefly explained.

Structure of this report is as following. First, data is loaded. Data is processed and a number of columns are added to calculate necessary features. Further, data is aggregated by each unique user. The next step will be the computation of features and explanation. Summary table of the resulted output will be given.

Data import and processing:

```
In [75]: import sys
         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         dateparse = lambda x: pd.datetime.strptime(x, '%Y-%m-%d %H:%M:%S')
         try:
             df_logs = pd.read_csv('logs.csv',
                                   parse_dates=['ts'],
                                   date_parser = dateparse,
                                   index_col=None)
         except FileNotFoundError:
             print('File not in current directory!')
             sys.exit(1)

         print(df_logs.shape)
```

```
(669491, 4)
```

Dataset consists of 4 columns and 669491 rows:

Column	Format	Description
uuid	string	user id

Column	Format	Description
ts	Datetime	Timestamp of log
useragent	string	Browser and OS info of user
hashed_ip	string	hashed ip address of user

Adding several columns to initial dataset in order to conveniently calculate neseccary features: Necessary columns for aggregation. First of all, a column consisting of only value of 1 is added for convenience. Column representing day of month is added later to calculate whether the user is a repeat visitor. Additionaly, day of week and hour columns are appended to calculate weekday_biz. Weekday business hours are considered 9 AM - 6 PM, Monday to Friday

```
In [78]: df_logs['highly_active'] = 1
         df_logs['day_of_month'] = df_logs['ts'].dt.day
         df_logs['day_of_week'] = df_logs['ts'].dt.dayofweek
         df_logs['hour'] = df_logs['ts'].dt.hour
         bs_hours, bs_days = range(9,19), range(1,6)
         df_logs['weekday_biz'] = (df_logs['hour'].isin(bs_hours)) \
         & (df_logs['day_of_week'].isin(bs_days))
```

Aggregating information by each distinct user User activity. Calculating each user's visit count distribution of which will a base to assign activity level.

```
In [79]: a = df_logs[['uuid','highly_active']].groupby('uuid').agg('sum')
```

Weekday business hours. For simplicity, the most occurring value is chosen. Meaning, value is True if the user visits the page during weekday business hours more frequently. Otherwise is False.

```
In [81]: b = df_logs[['uuid','weekday_biz']].\
         groupby('uuid').agg(lambda x:x.value_counts().index[0])
```

my_feature was chosen based on hashed_ip. Since it is hashed it not possible to detect user location. However, it is possible to calculate whether user accesses the website from multiple ip addresses. The logic is: if user has more than one ip address then the value is True. One of possible explanations to having multiple ip address might be that the user is a frequent traveller.

```
In [62]: c = df_logs[['uuid','hashed_ip']].groupby('uuid').agg({'hashed_ip': \
                                                                 pd.Series.nunique})
         c = c.T.squeeze()
         c = c.rename('my_feature')
```

Multiple days feature was calculating based on the count of unique days of month the user visits the page. Even if the user has a number of visits during the same day the value equals to False.

```
In [63]: d = df_logs.groupby('uuid')['day_of_month'].nunique()
         d = d.rename('multiple_days')
```

Series with aggregated information are added into a pandas dataframe.

```
In [74]: output = pd.concat([a,d, b,c], axis=1).reset_index()
```

Dataframe is almost ready. Only a few adjustments are needed. First of all, distribution of activity level need to be understood. It is obvious from quintile table that most of the users are one-time visitors. This is why 75% (equals to 2) quintile will be determined as a threshold for high activity. Meaning that a highly active visitor is the user visit count is in the top 25%.

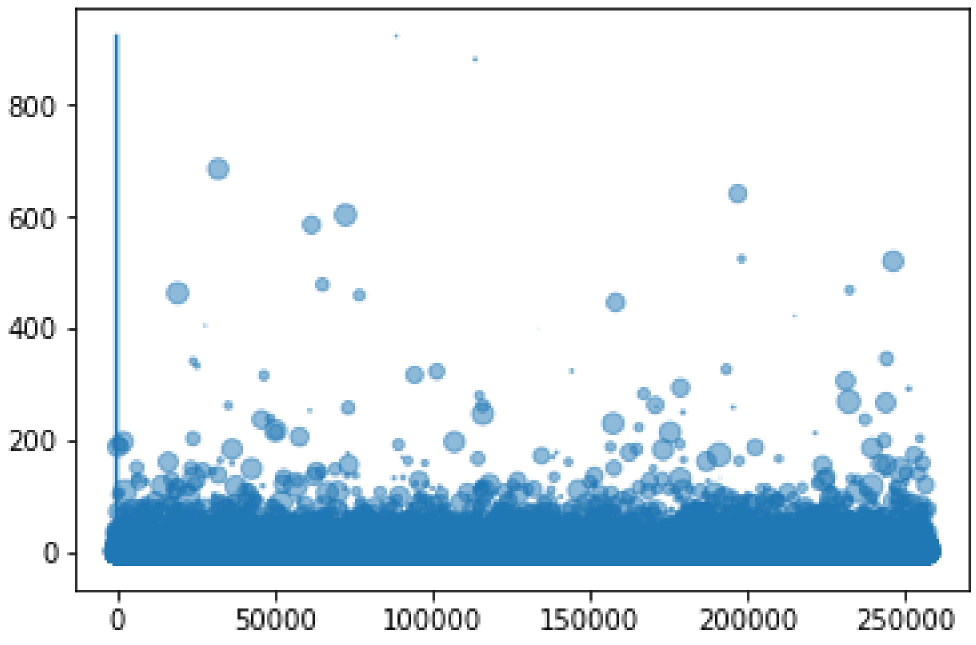
```
In [82]: output['highly_active'].describe()
         qs = [0.1,0.5,0.7,0.75,0.8,0.9,1]
         quan_table = output['highly_active'].quantile(qs)
         print(quan_table)

0.10      1.0
0.50      1.0
0.70      1.0
0.75      2.0
0.80      2.0
0.90      5.0
1.00    923.0
Name: highly_active, dtype: float64
```

Graph below shows distribution of user activeness.

```
In [66]: N= len(output['highly_active'])
         area = np.pi * (5 * np.random.rand(N))**2  # 0 to 15 point radii
         %matplotlib inline
         plt.plot(quan_table)
         plt.scatter(range(0, N), output['highly_active'], s=area, alpha=0.5)

Out[66]: <matplotlib.collections.PathCollection at 0x7ff154867f60>
```



Transforming **highly_active** feature values into True/False vector

```
In [67]: threshold = output['highly_active'].quantile(0.75)
         output['highly_active'] = np.where(output['highly_active']>=threshold,
                                            True, False)
```

Transforming **multiple_days, my_feature** column into True/False vector

```
In [68]: output['multiple_days'] = np.where(output['multiple_days']>1,
                                            True,False)
         output['my_feature'] = np.where(output['my_feature']>1,
                                         True,False)
```

Summary of the final table

```
In [69]: output[['highly_active',
                 'multiple_days',
                 'weekday_biz',
                 'my_feature']].apply(pd.Series.value_counts)

Out[69]:        highly_active  multiple_days  weekday_biz  my_feature
         False         185374         221319       159320      238272
         True           71980          36035        98034       19082
```

Table demonstrates that there are 185374 highly active customers

```
In [70]: pd.crosstab(output['highly_active'],columns=output['multiple_days'],
                     normalize='all')
```

```
Out[70]: multiple_days     False     True
         highly_active
         False          0.720307  0.000000
         True           0.139671  0.140021

In [71]: pd.crosstab(output['highly_active'],columns=output['weekday_biz'],
                     normalize='all')

Out[71]: weekday_biz      False     True
         highly_active
         False          0.46044  0.259868
         True           0.15863  0.121063

In [72]: pd.crosstab(output['highly_active'],columns=output['my_feature'],
                     normalize='all')

Out[72]: my_feature        False     True
         highly_active
         False          0.720307  0.000000
         True           0.205546  0.074147

In [73]: print( output.corr())

               highly_active  multiple_days  weekday_biz  my_feature
highly_active       1.000000       0.647547     0.066613    0.454144
multiple_days       0.647547       1.000000     0.058619    0.665772
weekday_biz         0.066613       0.058619     1.000000    0.018906
my_feature          0.454144       0.665772     0.018906    1.000000
```

Correlation matrix suggests that there is a positive correlation between highly_active, multiple_days and my_feature