A Data Science Test

Pedro Fernández Soler

February 22, 2020

Contents

1 Technical question

1.1 Statement of the problem

Build a script (R, Python, SQL...) to create a cohort of Signup to First Order and show the result. The objective of this cohort is to see, out of the users that signed up in Week N, how many did their first order in Week N+1, N+2, N+3... As an input, you have 2 tables : - one with user_id, signup_timestamp - and a second one with user_id, order_timestamp

1.2 Answer:

I have used Python3.7 to adress the problem, in combination with packages *numpy* and *pandas*. The example presented here assumes the existence of two pandas dataframes:

- dfSignUp: contains the columns 'user_id' and 'signup_timestamp'
- dfOrder: contains the columns 'user_id' and 'order_timestamp'

In order to test the script I have generated these two dataframes simulating a number of 600 unique 'user_id' and 'signup_timestamp', as well as 1200 random orders of these users population at random 'order_timestamp' (of course, the 'order_timestamp' were generated consistently with the 'signup_timestamp' of each user).

The sign ups where simulated in the period between Jan-May of 2018. The orders in a period of 35 days after the corresponding signup date, with smaller probabilities for the last days (with the purpose of testing if this pattern manifests in the cohort table).

The resulting cohort table is shown in Table 1. The code and auxiliary functions that I have written to generate the dataframes can be found in GitHub

https://github.com/peferso/CohortSignUpFirstOrder

	number	FO N+0 [%]	FO N+1 [%]	FO N+2 [%]	FO N+3 [%]	FO N+4 [%]
SU N=1	26	34.62	46.15	7.69	3.85	7.69
SU N=2	24	20.83	54.17	20.83	4.17	0.00
SU N=3	34	44.12	29.41	14.71	2.94	8.82
SU N=4	29	41.38	37.93	17.24	3.45	0.00
SU N=5	36	36.11	38.89	13.89	8.33	2.78
SU N=6	40	40.00	35.00	20.00	5.00	0.00
SU N=7	31	45.16	25.81	16.13	6.45	6.45
SU N=8	29	31.03	31.03	27.59	6.90	3.45
SU N=9	37	40.54	32.43	21.62	5.41	0.00
SU N=10	32	34.38	37.50	12.50	15.62	0.00
SU N=11	30	33.33	50.00	3.33	10.00	3.33
SU N=12	32	43.75	34.38	6.25	15.62	0.00
SU N=13	27	33.33	37.04	14.81	11.11	3.70
SU N=14	29	34.48	34.48	17.24	10.34	3.45
SU N=15	29	41.38	34.48	17.24	6.90	0.00
SU N=16	32	40.62	40.62	15.62	3.12	0.00
SU N=17	29	41.38	37.93	13.79	6.90	0.00

Table 1: Cohort to Signup to First Order. SU and FO stand for signup and first order respectively.

● The script ●

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
# script to create the cohort of Signup to First Order

# (*) STRATEGY: add a column to the dataframe dfOrder with the signup timestamp of each
   ID.
# (*) After that, everything reduces to operating with timestamps and counting
   occurrences

# First of all, I set a date in which weeks will start counting
init_date = "2018-01-01"
end_date = "2019-01-01"

# In table Orders, for each ID I add a column with the time stamp of the sign up
df = dfOrder # create a new table from the Orders table
nrows=df.shape[0]
ncols=df.shape[1]
list_sgnupdates = []
for row in range(nrows): # for each row
   dfaux = dfSignUp.loc[dfSignUp['user_id'] == dfOrder.iloc[row, 0]] # I search for the
      signup date of the ID
   sgdate = dfaux.iloc[0, 1] # get the value
   list_sgnupdates.append(sgdate) # I add an element to the list with the sign up time
      stamp of the row's ID
df['signup_timestamp'] = list_sgnupdates # add the column to the data frame
```

```
# I keep the orders of users which signed up in the year 2018
condition = (df['signup_timestamp'] >= init_date) & (df['signup_timestamp'] < end_date)
df = df.loc[condition]

# I add a column with the week in the year of the order, and the week in the year of the
    sign up
# I also add a column with the difference btw week of order and week of sgnup
df['week_signup_timestamp'] = pd.to_datetime(df['signup_timestamp']).dt.week
df['week_order_timestamp'] = pd.to_datetime(df['order_timestamp']).dt.week
df['week_order_dif'] = df['week_order_timestamp'] - df['week_signup_timestamp']

print('\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n' +
     '\nWe add a column to each order with\n' +
     'the signup time stamp of the user in the row.\n' +
     'We further add a column with the week in the\n' +
     'year of each time stamp and another with the difference:\n')
print(df.head(10))

# We order the rows by user_id and order_timestamp, keep only the columns with week data
cols = df.columns.tolist()
cols = [cols[0]] + cols[-1:-4:-1] #+ cols[1:2+1]
df = df[cols]
df = df.sort_values(["user_id", "week_order_timestamp"], axis=0, ascending=True)
print('\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n' +
     '\nWe arrange the rows ordered by user id \n' +
     'and by week of order time stamp to keep the first order.\n')
print(df.head(10))

# We keep the first occurrence in user_id to pick up the first order of each user
df = df.drop_duplicates(subset=['user_id'], keep='first')
print('\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n' +
     '... and keep only first occurrences in user_id. We further \n' +
     'add a column with the week of first order counting from week\n' +
     'of sign up                                         \n')
print(df.head(10))

# Now I can create the cohort
rownames = []
colnames = []
maxSUweek = df['week_signup_timestamp'].max()
maxFOweek = df['week_order_dif'].max()
# matrix of occurrence counting
# indices go to +1 because row 0 will no be used and col 0 will keep total occurrences
    in row
occ_matrix = np.zeros((maxSUweek+1, maxFOweek+1), dtype=np.single)

# the labels of the cohort
colnames.append('number')
for week_sgnUp in range(1, maxSUweek + 1):
    rownames.append("SU N=" + str(week_sgnUp))
```

```
for week_FO in range(maxFOweek):
    colnames.append("FO N+" + str(week_FO)+" [%]")

# we compute the cohort checking number of occurrences for each entry
# and then transforming it to percentage dividing by total number
# of occurrences in each row
for week_sgnUp in range(1, maxSUweek + 1):
    for week_FO in range(1, maxFOweek + 1):
        condition = (df['week_signup_timestamp'] == week_sgnUp) & (df['week_order_dif']
            == week_FO - 1)
        occ_matrix[week_sgnUp, week_FO] = len(df[condition])
    rowsum = np.sum(occ_matrix[week_sgnUp, 1:], dtype=np.int32)
    occ_matrix[week_sgnUp, 0] = rowsum
    occ_matrix[week_sgnUp, 1:] = occ_matrix[week_sgnUp, 1:]/rowsum*100.0
    print(np.round(occ_matrix[week_sgnUp, :], 2))

# build the dataframe
dict_cohort = {}
icol = 0
for i in colnames:
    dict_cohort[i] = list(occ_matrix[1:, icol])
    icol = icol + 1

cohort = DataFrame(dict_cohort, columns=colnames, index=rownames)
# we round the entries
cohort = cohort.round(2)
# and include the number of occurrences in each row as an integer
cohort[colnames[0]] = pd.to_numeric(cohort[colnames[0]], downcast='integer')

# we see the result
print(cohort.to_string())
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
```