

Deliverable 3

Vibons - Journey App - Team 1

Group members: Jiaqi Zhao, Lingyan Jiang, Ruoqi Shi

CS506 Computational Tools of Data Science

Instructor: Lance Galletti

Client: Vibons Journey App

Project Manager: Shubhangi Jain

Table of Contents

1. Motivation	3
2. Dataset and Representation	4
3. Data Preprocessing	5
4. Technology Approach	6
Iteration 1:	6
Aim:	6
Strategy, Result and Visualization:	6
Insights:	7
Pitfalls and Challenge:	8
Iteration 2:	9
Aim:	9
Strategy, results and visualizations	9
Lessons Learned and Insights:	13
Pitfalls and Challenge:	14
Iteration 3:	15
Aim:	15
Strategy, Result and Visualizations:	15
Lessons Learned and Insights:	19
Pitfalls and Future steps:	19
5. Evaluations & Questions for our client	21
6. Reference	22

1. Motivation

Journey App, designed by Vibons, is a motivational app for users to easily jot down their daily ideas, write down their daily emotions, and receive educational articles about customized topics through notifications. Nevertheless, as every user can be inundated with so many app notifications, our client wants to find an optimal time to notify the users of the contents and inspire them to read.

In a nutshell, our primary goal is to find the send time optimization for each user of the Journey App, based on the dataset given by our clients. Send time optimization analyzes when is the most likely time for each recipient to open the notifications. Such an analysis can boost the open rates for our client and enhance engagement with all the users. Our secondary goal is to build a model to predict optimal notification sending time for future users.

2. Dataset and Representation

For all the features, we knew their meanings before we preprocessed the data. For the definition of data, since each company has its own idea of formulating data titles, the meaning of data may be completely different. Therefore, we ask our customers in detail. We will also add the explanation of the data labels in the README file so that people can understand clearly our data.

- Info Customer ID - The ID number of the users' company, there are about 20 company IDs in total.
- User Id - Each user's Id on behalf of which notifications are sent.
- User-Created At - The first date of hire of the user to whom notification is sent.
- Activation Date - Date and time when a notification is sent.
- Activity Date - Date and time when a notification is opened by the receiving user.
- Name - The title of the notification content.
- Content ID - the ID of the notification content name.
- Content Type - The type of notification content.
- Journey Name - The journey that the content of the notification comes from.
- Action - Percentage of content read by the end user.
- Device - The device used by the user to read the notification. Due to the current universality and convenience of mobile phones, the focus is on mobile phone users.
- Channel - The channel that users click on the notification.
- Session Id: When a notification is sent to the user, a session is locked in the system.

3. Data Preprocessing

After loading the CSV file as dataframe, we first checked about NaN values inside our dataframe:

```
: 1 X.isna().sum()
: [2020-10-25 14:40:25] production_jobs.INFO: Customer Id      0
  User Id                                                    2033
  User Created At                                           2197
  Activation Date                                           2033
  Activity Date                                             2033
  Name                                                       2033
  Content Type                                              4124
  Content Id                                                4124
  Journey Name                                              4124
  Action                                                    4124
  Duration                                                  20918
  Device                                                    6215
  Channel                                                    6215
  Session Id                                                6685
  Rating                                                    6215
  dtype: int64
```

For content type, device, and channel, they can be categorical variables. So we converted them into categorical variables with integers starting from one and if we detect a NaN value, we label it as zero.

Then, we use `fillna()` method to replace all the NaN values with the mode corresponding to each variable. The reason why we fit the mode value here is that we want the tendency of the overall users but we do not want outliers (if exist) to influence the data. We use `dropna()` method to drop the row if there is a NaN value for that user.

At last, we delete “FLIPBOOK” entries inside the column “User Id”.

4. Technology Approach

Iteration 1:

Aim:

In order to find the optimal notification sending time, we use the activity date column to parse out the hour and find out the hour time with the highest frequency number and stored in a dictionary.

Strategy, Result and Visualization:

1. Initial Stage:

We created a new dataframe called open_time with columns of User Id, Activation Date, and Activity Date from clean_data.

	User Id	Activation Date		Activity Date	
0	147	2018-09-25	21:05:24	2018-09-27	09:30:53
1	139	2018-09-22	10:00:02	2018-10-01	11:18:20
2	139	2018-09-23	10:00:59	2018-10-01	11:19:18
3	200	2018-10-01	08:00:36	2018-10-03	09:48:55
4	200	2018-10-01	09:00:52	2018-10-03	10:38:28
...	
1796933	44365	2020-09-25	16:00:00	2020-10-08	23:12:58
1796934	43068	2020-09-25	14:30:00	2020-10-08	23:13:04
1796935	43068	2020-10-08	16:00:00	2020-10-08	23:13:23
1796936	44365	2020-09-25	16:00:00	2020-10-08	23:13:26
1796937	44365	2020-09-21	16:00:00	2020-10-08	23:13:33

2. Developing Stage 1:

By this dataframe, we will be able to find out the time lag between activity date and activation date by subtracting activation date from activity date, which is the time lag between the date and time when a notification is sent and when a notification is opened by the receiving user. The result is stored in a new column Time lag in open_time data

frame shown below:

	User Id	Activation Date	Activity Date	Time lag
0	147	2018-09-25 21:05:24	2018-09-27 09:30:53	1 days 12:25:29
1	139	2018-09-22 10:00:02	2018-10-01 11:18:20	9 days 01:18:18
2	139	2018-09-23 10:00:59	2018-10-01 11:19:18	8 days 01:18:19
3	200	2018-10-01 08:00:36	2018-10-03 09:48:55	2 days 01:48:19
4	200	2018-10-01 09:00:52	2018-10-03 10:38:28	2 days 01:37:36
...
1796933	44365	2020-09-25 16:00:00	2020-10-08 23:12:58	13 days 07:12:58
1796934	43068	2020-09-25 14:30:00	2020-10-08 23:13:04	13 days 08:43:04
1796935	43068	2020-10-08 16:00:00	2020-10-08 23:13:23	0 days 07:13:23
1796936	44365	2020-09-25 16:00:00	2020-10-08 23:13:26	13 days 07:13:26
1796937	44365	2020-09-21 16:00:00	2020-10-08 23:13:33	17 days 07:13:33

3. Developing Stage 2:

- We want to create a dictionary for hr item of each user so that we can give the optimal sending time by observing the frequency of hr. For here, we use the first 10 users as an example. By importing datetime.strptime library, we can easily parse out the hour from the activity date.
- Then, we store the user id as key and all the corresponding hours as a list for the value of the key. Next, we want to find the frequency for each user. We use Counter() and most_common() function from collections library. For each user, we get the first two highest frequency hours for each user id by most_common(). This method gives us a result of all the hour of a certain user with the frequency of hour as value in a dictionary ranking by the highest frequency to the lowest one.
- By setting the number of result to be shown as 2, only the first 2 hour will be shown. If there's only one hour, we will store it directly to our dictionary. If one of the hour is higher than the another one, we will only store the highest one. If the two frequency are the same, we will save both hour time towards the user id as a list in the dictionary. By this method, the result of first ten users are shown below:

```
1 print(dic_frequency)
```

```
{'1': 12, '2': 20, '21': [8, 15], '22': 11, '38': 3, '41': 10, '47': 14, '53': 13, '61': 16, '62': 11}
```

Insights:

In this iteration, we used a dictionary to find out the corresponding hour time with the highest frequency for each user. The reason why we are using a dictionary is that it is easy for the client to find out the corresponding hour time for each user by searching the user id.

Pitfalls and Challenge:

First of all, though a dictionary is easy for our client to query the sending time for each user, we did not pay attention to the high running time of the dictionary. It takes us a really long time to generate the result. Thus, it is important for us to find out a new data structure to store and run the algorithm. We choose dataframe from the pandas library. The built-in functions inside this library can reduce and avoid the use of for loops which can save lot's of running time.

Secondly, we did not categorize the hour time with different week days. It is pivotal to find out the optimal sending time with the corresponding day of the week. We should not consider the seven days as a whole because our client might need to send several notifications in a week. We should give our client which day of the week and the sending time together.

Thirdly, in this iteration, we only consider the highest one. From our result, there's a case that the `most_common(2)` gives as two hour times with the same frequency. In the future iterations, we need to increase the number chosen for `most_common()` function. Or we might need a new criteria to choose the top nth frequency for each user.

Iteration 2:

Aim:

To find the optimal notification sending time based on the existing the “Activity date” column (when the notification was opened) and “Action” column (the completion rate when an user opens the notification) in the dataset of the existing users

Strategy, results and visualizations

1. Initial stage:
 - a. Based on our client’s requirement, we deleted the “time difference” and “time lag” which we created in Iteration 1.
 - b. We focused on the “activity date” and “action rate”
2. Data preprocessing:
 - a. First of all, we converted each date into the corresponding weekday of that week, as shown in the ‘Activity Day’ column
 - b. We then extracted the hour time of the notification open time for each row, as shown in the ‘hour’ column.

	User Id	Activation Date	Activity Date	Name	Action	User Created At	Activity Day	hour
0	217	2018-10-11 16:15:13	2018-10-13 23:06:41	LMS tarihe karşiyor: LXP dünyasına hoş geldiniz!	100	2018-10-09 13:37:22	Saturday	23
1	217	2018-10-11 11:52:48	2018-10-13 23:06:46	Silikon Vadisinden en son İK ve Eğitim Trendleri	100	2018-10-09 13:37:22	Saturday	23
2	217	2018-10-06 10:00:06	2018-10-13 23:06:58	Drucker'a göre İnovasyon Fırsatı Sağlayan Yedi...	100	2018-10-09 13:37:22	Saturday	23
3	217	2018-10-05 08:00:34	2018-10-15 14:30:20	8 maddede yıkıcı inovasyon	100	2018-10-09 13:37:22	Monday	14
4	217	2018-10-05 08:00:34	2018-10-15 14:30:49	8 maddede yıkıcı inovasyon	100	2018-10-09 13:37:22	Monday	14

Take the first row as an example. The ‘hour’ column for the first row is ‘23’. This means that user 217 opened this particular notification at 23:00, i.e. 11 pm, on October 13th, 2018 (as shown in the ‘Activity Date’ column of the first row)

- c. According to the day of the week, we converted it into an integer stored in a column called hour_num. For instance, Monday was converted into 100; Tuesday was converted into 200. After that, each hour was stored as an integer in a column called day_num. So it is easy to do calculations and statistics compared with a string.
- d. Then, we added day_num and hour_num together as a new column called total_num.

	User Id	Activation Date	Activity Date	Name	Action	User Created At	Activity Day	hour	day_num	total_num
0	217	2018-10-11 16:15:13	2018-10-13 23:06:41	LMS tarihe karşiyor: LXP dünyasına hoş geldiniz!	100	2018-10-09 13:37:22	Saturday	23	600	623
1	217	2018-10-11 11:52:48	2018-10-13 23:06:46	Silikon Vadisinden en son İK ve Eğitim Trendleri	100	2018-10-09 13:37:22	Saturday	23	600	623
2	217	2018-10-06 10:00:06	2018-10-13 23:06:58	Drucker'a göre İnovasyon Fırsatı Sağlayan Yedi...	100	2018-10-09 13:37:22	Saturday	23	600	623
3	217	2018-10-05 08:00:34	2018-10-15 14:30:20	8 maddede yıkıcı inovasyon	100	2018-10-09 13:37:22	Monday	14	100	114
4	217	2018-10-05 08:00:34	2018-10-15 14:30:49	8 maddede yıkıcı inovasyon	100	2018-10-09 13:37:22	Monday	14	100	114

Take the first row of the screenshot above as an example, the ‘total_num’ column is ‘623’. This means that the user opens a notification on Saturday (‘6’) at 11 pm (‘23’).

- e. Next, we aggregated each row by user id. The user id became the index of each row.

User Id	Activation Date	Activity Date	Name	Action	User Created At	Activity Day	hour	day_num	total_num
21	[2018-10-31 13:32:19, 2018-11-01 08:30:57, 201...	[2018-10-31 14:08:53, 2018-11-01 08:33:27, 201...	[Çalışan Bağlılığı 3.0: Kişiyi Özel "Nudge (Dü...	[100, 100, 100, 100, 100, 100, ...	[2018-05-14 02:49:06, 2018-05-14 02:49:06, 201...	[Wednesday, Thursday, Thursday, Wedn...	[14, 8, 8, 8, 11, 9, 15, 15, 15, 15, 16, 16, 8]	[300, 400, 400, 400, 300, 400, 300, 300, 300, ...	[314, 408, 408, 408, 311, 409, 315, 315, 315, ...]
38	[2019-10-03 08:00:00, 2019-10-03 09:00:00]	[2019-10-04 03:12:57, 2019-10-04 03:13:10]	[demo flip, safety]	[100, 100]	[2018-05-14 17:59:52, 2018-05-14 17:59:52]	[Friday, Friday]	[3, 3]	[500, 500]	[503, 503]
41	[2019-02-21 16:00:00, 2019-02-21 16:39:00, 201...	[2019-02-22 08:16:24, 2019-02-22 08:17:25, 201...	[Makas, Hakan Ateş'ten Denizcilerimize Mesajla...	[100, 100, 100, 100, 100, 100, ...	[2018-05-16 19:43:48, 2018-05-16 19:43:48, 201...	[Friday, Friday, Friday, Friday, Friday, Monda...	[8, 8, 11, 14, 15, 14, 15, 15, 15, 9, 10, 23, ...	[500, 500, 500, 500, 500, 100, 400, 100, 100, ...	[508, 508, 511, 514, 515, 114, 415, 115, 115, ...]
47	[2020-10-07 12:45:00, 2020-10-07 12:45:00, 202...	[2020-10-07 14:27:21, 2020-10-07 14:27:22, 202...	[Ekipleri Uzaktan Etkili Yönetmek, Ekipleri Uz...	[100, 100, 100]	[2018-06-21 08:01:03, 2018-06-21 08:01:03, 201...	[Wednesday, Wednesday, Wednesday]	[14, 14, 14]	[300, 300, 300]	[314, 314, 314]
53	[2018-11-01 08:30:57, 2018-11-08 08:30:00, 201...	[2018-11-17 13:04:53, 2018-11-17 13:06:15, 201...	[Çalışan Bağlılığı 3.0: Kişiyi Özel "Nudge (Dü...	[100, 100, 100]	[2018-06-27 07:53:16, 2018-06-27 07:53:16, 201...	[Saturday, Saturday, Saturday]	[13, 13, 13]	[600, 600, 600]	[613, 613, 613]

The screenshot above shows the result after aggregating rows with the same user id. Take the first row of the screenshot above as an example. The ‘total_num’ column becomes a list of [315, 408, 408, 408, 311, 409, 315, 315, 315, ...]. This list shows when and what time did this user 21 opened the notifications in the whole dataset.

- f. Then, we went through the total number and found out the frequency/impression (numbers of times) of opening the notification at a specific hour time for each user on each day of the week. Our algorithm gave a list with seven elements corresponding from Monday to Sunday with the hour time and frequency as a list in it.

User Id	Activation Date	Activity Date	Name	Action	User Created At	Activity Day	hour	day_num	total_num	count_total	every_day_freq
21	[2018-10-31 13:32:19, 2018-11-01 08:30:57, 201...	[2018-10-31 14:08:53, 2018-11-01 08:33:27, 201...	[Çalışan Bağlılığı 3.0: Kişiyi Özel "Nudge (Dü...	[100, 100, 100, 100, 100, 100, ...	[2018-05-14 02:49:06, 2018-05-14 02:49:06, 201...	[Wednesday, Thursday, Thursday, Thursday, Wedn...	[14, 8, 8, 8, 11, 9, 15, 15, 15, 16, 16, 8]	[300, 400, 400, 400, 300, 400, 300, 300, 300, 300, ...	[314, 408, 408, 408, 311, 409, 315, 315, 315, ...	[[315, 4), (408, 3), (116, 2), (208, 1), (315, 4), (408, 3), (311, ...]	
	[2019-10-03 08:00:00, 2019-10-03 09:00:00]	[2019-10-04 03:12:57, 2019-10-04 03:13:10]	[demo flip, safety]	[100, 100]	[2018-05-14 17:59:52, 2018-05-14 17:59:52]	[Friday, Friday]	[3, 3]	[500, 500]	[503, 503]	[[503, 2)]	[[[0, 0, 0, 0, 0, 0, 0], [503, 2), 0, 0]]
	[2019-02-21 16:00:00, 2019-02-21 16:39:00, 201...	[2019-02-22 08:16:24, 2019-02-22 08:17:25, 201...	[Makas, Hakan Ateş'ten Denizcilerimize Mesajla...	[100, 100, 100, 100, 100, 100, ...	[2018-05-16 19:43:48, 2018-05-16 19:43:48, 201...	[Friday, Friday, Friday, Friday, Friday, Monda...	[8, 8, 11, 14, 15, 15, 15, 9, 10, 23, ...	[500, 500, 500, 500, 500, 100, 400, 100, ...	[508, 508, 511, 514, 515, 114, 415, 115, ...]	[[410, 63), (309, 17), (111, 14), (416, 14), (...]	
	[2020-10-07 12:45:00, 2020-10-07 12:45:00, 202...	[2020-10-07 14:27:21, 2020-10-07 14:27:22, 202...	[Ekipleri Uzaktan Etkili Yönetmek, Ekipleri Uz...	[100, 100, 100]	[2018-06-21 08:01:03, 2018-06-21 08:01:03, 201...	[Wednesday, Wednesday, Wednesday]	[14, 14, 14]	[300, 300, 300]	[314, 314, 314]	[[314, 3)]	[[[0, 0, 0, 0, 0, 0, 0], [314, 3), 0, 0, 0, 0, 0]]
53	[2018-11-01 08:30:57, 2018-11-08 08:30:00, 201...	[2018-11-17 13:04:53, 2018-11-17 13:06:15, 201...	[Çalışan Bağlılığı 3.0: Kişiyi Özel "Nudge (Dü...	[100, 100, 100]	[2018-06-27 07:53:16, 2018-06-27 07:53:16, 201...	[Saturday, Saturday, Saturday]	[13, 13, 13]	[600, 600, 600]	[613, 613, 613]	[[613, 3)]	[[[0, 0, 0, 0, 0, 0, 0], [613, 3), 0]]

3. Developing Stage 1 :

- a. Goal: We focused on the users who have a completion rate of 100 and found the hour time on each day of the week for each user when they open the notification and complete 100% of the contents.
- b. Steps:
 - i. Subtract all the users who have a completion rate of “100”
 - ii. Under the condition of completion rate equal to “100”, create a list of the days and the hour time about when they opened the notification for each user, as shown in the ‘total_num’ column in the screenshot below:

Out[251]:

User Id	Activation Date	Activity Date	Name	Action	User Created At	Activity Day	hour	day_num	total_num
21	[2018-10-31 13:32:19, 2018-11-01 08:30:57, 201...	[2018-10-31 14:08:53, 2018-11-01 08:33:27, 201...	[Çalışan Bağlılığı 3.0: Kişiye Özel “Nudge (Dü...	[100, 100, 100, 100, 100, 100, 100, ...	[2018-05-14 02:49:06, 2018-05-14 02:49:06, 201...	[Wednesday, Thursday, Thursday, Thursday, Wedn...	[14, 8, 8, 8, 11, 9, 15, 15, 15, 15, 16, 16, 8]	[300, 400, 400, 400, 300, 400, 300, 300, 300, ...	[314, 408, 408, 408, 311, 409, 315, 315, 315, ...
38	[2019-10-03 08:00:00, 2019-10-03 09:00:00]	[2019-10-04 03:12:57, 2019-10-04 03:13:10]	[demo flip, safety]	[100, 100]	[2018-05-14 17:59:52, 2018-05-14 17:59:52]	[Friday, Friday]	[3, 3]	[500, 500]	[503, 503]
41	[2019-02-21 16:00:00, 2019-02-21 16:39:00, 201...	[2019-02-22 08:16:24, 2019-02-22 08:17:25, 201...	[Makas, Hakan Ateş’ten Denizcilerimize Mesajla...	[100, 100, 100, 100, 100, 100, ...	[2018-05-16 19:43:48, 2018-05-16 19:43:48, 201...	[Friday, Friday, Friday, Friday, Friday, Monda...	[8, 8, 11, 14, 15, 14, 15, 15, 15, 9, 10, 23, ...	[500, 500, 500, 500, 500, 100, 400, 100, 100, ...	[508, 508, 511, 514, 515, 114, 415, 115, 115, ...
47	[2020-10-07 12:45:00, 2020-10-07 12:45:00, 202...	[2020-10-07 14:27:21, 2020-10-07 14:27:22, 202...	[Ekipleri Uzaktan Etkili Yönetmek, Ekipleri Uz...	[100, 100, 100]	[2018-06-21 08:01:03, 2018-06-21 08:01:03, 201...	[Wednesday, Wednesday, Wednesday]	[14, 14, 14]	[300, 300, 300]	[314, 314, 314]
53	[2018-11-01 08:30:57, 2018-11-08 08:30:00, 201...	[2018-11-17 13:04:53, 2018-11-17 13:06:15, 201...	[Çalışan Bağlılığı 3.0: Kişiye Özel “Nudge (Dü...	[100, 100, 100]	[2018-06-27 07:53:16, 2018-06-27 07:53:16, 201...	[Saturday, Saturday, Saturday]	[13, 13, 13]	[600, 600, 600]	[613, 613, 613]

(In the ‘total_num’ column, the number “314”, for example, means that User 21 opened the notification on Wednesday-represented as “3”, at 14:00, or 2 pm.)

- iii. List out the frequency/impression (numbers of times) of opening the notification at a specific hour time for each user on each day of the week, as shown in the ‘every_day_freq’ column in the screenshot below:

Out[370]:

User Id	Activation Date	Activity Date	Name	Action	User Created At	Activity Day	hour	day_num	total_num	count_total	every_day_freq
21	[2018-10-31 13:32:19, 2018-11-01 08:30:57, 201...	[2018-10-31 14:08:53, 2018-11-01 08:33:27, 201...	[Çalışan Bağlılığı 3.0: Kişiye Özel “Nudge (Dü...	[100, 100, 100, 100, 100, 100, 100, ...	[2018-05-14 02:49:06, 2018-05-14 02:49:06, 201...	[Wednesday, Thursday, Thursday, Thursday, Wedn...	[14, 8, 8, 8, 11, 9, 15, 15, 15, 15, 16, 16, 8]	[300, 400, 400, 400, 300, 400, 300, 300, 300, ...	[314, 408, 408, 408, 311, 409, 315, 315, 315, ...	[(315, 4), (408, 3), (408, 2), (315, 4), (314, 1), (311, ...	[[[(116, 2)], [(208, 1)], [(315, 4)], [(408, 3)]]
38	[2019-10-03 08:00:00, 2019-10-03 09:00:00]	[2019-10-04 03:12:57, 2019-10-04 03:13:10]	[demo flip, safety]	[100, 100]	[2018-05-14 17:59:52, 2018-05-14 17:59:52]	[Friday, Friday]	[3, 3]	[500, 500]	[503, 503]	[(503, 2)]	[[[0, 0, 0, 0], [(503, 2)], [0, 0]]
41	[2019-02-21 16:00:00, 2019-02-21 16:39:00, 201...	[2019-02-22 08:16:24, 2019-02-22 08:17:25, 201...	[Makas, Hakan Ateş’ten Denizcilerimize Mesajla...	[100, 100, 100, 100, 100, 100, ...	[2018-05-16 19:43:48, 2018-05-16 19:43:48, 201...	[Friday, Friday, Friday, Friday, Friday, Monda...	[8, 8, 11, 14, 15, 14, 15, 15, 15, 9, 10, 23, ...	[500, 500, 500, 500, 500, 100, 400, 100, 100, ...	[508, 508, 511, 514, 515, 114, 415, 115, 115, ...	[(410, 63), (309, 17), (111, 14), (210, 9), (215, 9), [(309, 17)]]	[[[(111, 14)], [(210, 9)], [(215, 9)], [(309, 17)]]
47	[2020-10-07 12:45:00, 2020-10-07 12:45:00, 202...	[2020-10-07 14:27:21, 2020-10-07 14:27:22, 202...	[Ekipleri Uzaktan Etkili Yönetmek, Ekipleri Uz...	[100, 100, 100]	[2018-06-21 08:01:03, 2018-06-21 08:01:03, 201...	[Wednesday, Wednesday, Wednesday]	[14, 14, 14]	[300, 300, 300]	[314, 314, 314]	[(314, 3)]	[[[0, 0, 0], [(314, 3)], [0, 0, 0]]
53	[2018-11-01 08:30:57, 2018-11-08 08:30:00, 201...	[2018-11-17 13:04:53, 2018-11-17 13:06:15, 201...	[Çalışan Bağlılığı 3.0: Kişiye Özel “Nudge (Dü...	[100, 100, 100]	[2018-06-27 07:53:16, 2018-06-27 07:53:16, 201...	[Saturday, Saturday, Saturday]	[13, 13, 13]	[600, 600, 600]	[613, 613, 613]	[(613, 3)]	[[[0, 0, 0, 0], [(613, 3)], [0, 0, 0]]

(In the 'every_day_freq' column, (116,2) means that User 21 opened the notification at 16:00, or 4pm, on Monday twice. For User 38, you can see some empty '[]' in it. This means that for other days of the week except Friday, this user did not 100% complete any article after opening the notifications.)

- iv. We then chose the hour time with the highest impression for each day as the optimal notification sending time for a particular user on a particular day of a week.
 - After the two steps above, we got the following result, shown in the screenshot.

Out[392]:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
User Id							
21	[16]	[8]	[15]	[8]	[nan]	[nan]	[nan]
38	[nan]	[nan]	[nan]	[nan]	[3]	[nan]	[nan]
41	[11]	[10, 15]	[9]	[10]	[9]	[17]	[12]
47	[nan]	[nan]	[14]	[nan]	[nan]	[nan]	[nan]
53	[nan]	[nan]	[nan]	[nan]	[nan]	[13]	[nan]

(For example, for User 21, we saved '16' for Monday. This means that we suggest sending the notification to User 21 at 16:00, i.e. 4pm. We chose '16' because, in the last screenshot, we found that '116' has the highest frequency for this user, where '1' indicates Monday and '16' indicates 16:00.)

4. Development Stage 2:

- a. Goal: To fill the 'nan' values in the results we got from the last stage for every user, which means to find the optimal, we will go through the time when each user opened the notification but with a different completion rate.
- b. Steps:
 - i. We first continued to distinguish between different daily best notification sending times when users are at different completion rates.
 - Here, in order to find out the situation of the completion rate and write code conveniently, we divided the completion rate into six buckets: 100 (analyzed in developing stage 1), 90-100, 80-90, 50-80, 0-50, 0.

- ii. We then repeated the steps in developing stage 1 for each bucket and found the best notification sending time each day under different completion rates
 - iii. When the best delivery time of the day was not found at completion rate=100, we chose to use completion rate = 90-100 instead, and so on until completion close to 0.
- c. Results:
Take the bucket when the completion rate equals to 90-100 as an example:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
User Id							
41	[15, 16, 11, 18]	[15]	[15, 16, 11, 18]	[nan]	[nan]	[nan]	[nan]
61		[12]	[nan]	[nan]	[nan]	[nan]	[nan]
62		[11]	[nan]	[nan]	[nan]	[nan]	[nan]
78		[nan]	[15]	[12, 15]	[23]	[11]	[15]
79		[23]	[19]	[13, 14]	[23]	[13]	[nan]

These are the first five users when their completion rate is less than 100. Take User 41 as an example. We saved a list of '15,16,11,18' for Monday. This means that we suggest sending the notification to User 41 on Monday at 15:00, 16:00, 11:00 and 18:00, i.e. 3 pm, 4 pm, 11 am and 6 pm. The 'nan' value here indicates that the user does not have a completion rate of 90-100 on the particular day of the week.

Lessons Learned and Insights:

First of all, we learned the importance of simplifying the elements when the dataset is pretty large. Our dataset is a quite large one with almost 2 million data entries, specifically 1926480 rows. Thus, for instance, it will take a long time for the algorithm to process if we are dealing with the weekday element in the format of string. So we creatively turned the weekday element from string to numbers, such as '100', as shown in the 'day_num' column. So it is much easier to do calculations and statistics compared with a string.

Second, we learned the importance of result demonstration. This means that sometimes for our client, they care more about the results regarding the primary goal of the project. They care less about what approaches or how did we do to get the results, especially when the client is not a technical person. Thus, it is important to output results that are readable and easy-to-understand, as we did in which we eliminated unnecessary columns and only showed the optimal notification sending time for each user on each day of the week.

Pitfalls and Challenge:

As for this iteration of the project, when picking the optimal notification sending time for each user on each day of the day, we only provided one choice based on which hour time has the largest impression/frequency. This approach may delete some valuable information when the largest and second to the largest impression/ frequency of the hour time are quite close to each other.

For instance, for user 21, on Monday, they opened the notification at 2 pm for 10 times, while at 6 pm for 9 times. The approach in this iteration only keeps 2 pm as the optimal notification sending time on Monday, while does not keep the 6 pm. However, 6 pm may also be valuable for the client to try out. Thus, we tried different ways to improve this issue in the next iteration.

Iteration 3:

Aim:

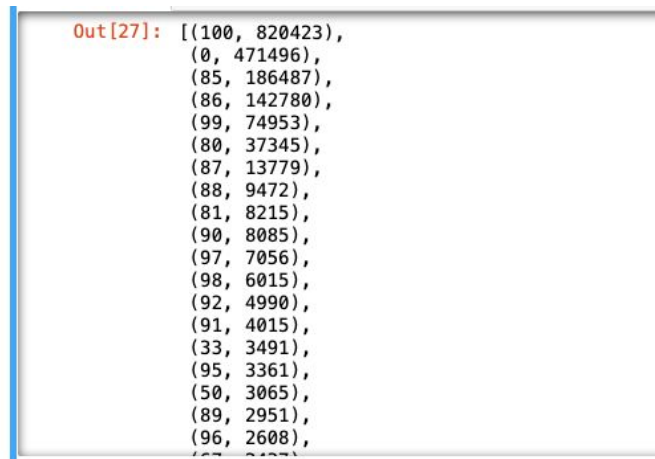
According to the client's request to add the customer impression into the final result, the algorithm of optimal notification sending time was changed again, and the format required by the client was changed to get a better result performance.

Strategy, Result and Visualizations:

After we met with our project manager and client, our client gave us some feedback and new request for this project. Instead of finding the optimal notification sending time for all application users, the client wanted our team to only focus on the users whose notification completion rate is between 85 and 100 and added a new request of considering the customer impression in our final result. In this case, our team needed to find different optimal times under different completion rates and then compare the completion rate to find the optimal time for each day. In addition we needed to consider the different content types and list the contents that users read at that optimal notification sending time.

1. Initial stage:

- a. Our team added the action distribution to see the distribution for different completion rate and users range. Then, we divided the users into three groups by the completion rate: 85-100, 0-85, and 0.



```
Out[27]: [(100, 820423),
          (0, 471496),
          (85, 186487),
          (86, 142780),
          (99, 74953),
          (80, 37345),
          (87, 13779),
          (88, 9472),
          (81, 8215),
          (90, 8085),
          (97, 7056),
          (98, 6015),
          (92, 4990),
          (91, 4015),
          (33, 3491),
          (95, 3361),
          (50, 3065),
          (89, 2951),
          (96, 2608),
          ...]
```

As you can see the screenshots above, our team first found out the completion rate distribution. It clearly shows that the users who have the completion rate =100 are in the majority. The following screenshot is the dataframe of all users whose completion rate between 85 and 100.

User Id	Activation Date	Activity Date	Name	Content Type	Action	User Created At	Activity Day	hour	day_num	total_num	Content_Type_total	
0	217	2018-10-11 16:15:13	2018-10-13 23:06:41	LMS tarihe karışıyor: LXP dünyasına hoş geldiniz!	11000	100	2018-10-09 13:37:22	Saturday	23	600	623	11623
1	217	2018-10-11 11:52:48	2018-10-13 23:06:46	Silikon Vadisinden en son İK ve Eğitim Trendleri	11000	100	2018-10-09 13:37:22	Saturday	23	600	623	11623
2	217	2018-10-06 10:00:06	2018-10-13 23:06:58	Drucker'a göre İnovasyon Fırsatı Sağlayan Yedi...	12000	100	2018-10-09 13:37:22	Saturday	23	600	623	12623
3	217	2018-10-05 08:00:34	2018-10-15 14:30:20	8 maddede yıkıcı inovasyon	14000	100	2018-10-09 13:37:22	Monday	14	100	114	14114
4	217	2018-10-05 08:00:34	2018-10-15 14:30:49	8 maddede yıkıcı inovasyon	14000	100	2018-10-09 13:37:22	Monday	14	100	114	14114

- b. We created five new dataframe columns for this dataframe and they are “Activity Day”, “hour”, “day_num”, “total_num”, and “Content_Type_total”. “Activity Day” is the date of each “Activity Date”. “Hour” is the hour time of each “Activity Date”. “Day_num” is the transformation of “Activity Day” in order to facilitate future statistics and calculations. “Total_num” is the result of adding “hour” and “day_num” in order to put the activity time of the same user in a space in the dataframe. “Content_Type_total” is the result of adding “total_num” and “Content Type” together so that we can easily report all the content that users read at each optimal notification sending time.

User Id	Activation Date	Activity Date	Name	Content Type	Action	User Created At	Activity Day	hour	day_num	total_num	Content_Type_total
21	[2018-10-31 13:32:19, 2018-11-01 08:30:57, 201...]	[2018-10-31 14:08:53, 2018-11-01 08:33:27, 201...]	[Çalışan Bağlılığı 3.0: Kişiyi Özel "Nudge (Dü...]	[11000, 11000, 12000, 14000, 14000, 140...]	[100, 100, 100, 100, 100, 100, ...]	[2018-05-14 02:49:06, 2018-05-14 02:49:06, 201...]	[Wednesday, Thursday, Thursday, Thursday, Wedn...]	[14, 8, 8, 8, 11, 9, 15, 15, 15, 16, 8]	[300, 400, 400, 400, 300, 400, 300, 300, 300, ...]	[314, 408, 408, 408, 311, 409, 315, 315, 315, ...]	[11314, 11408, 11408, 12408, 14311, 14409, 143...]
38	[2019-10-03 08:00:00, 2019-10-03 09:00:00]	[2019-10-04 03:12:57, 2019-10-10 03:13:10]	[demo flip, safety]	[12000, 14000]	[100, 100]	[2018-05-14 17:59:52, 2018-05-14 17:59:52]	[Friday, Friday]	[3, 3]	[500, 500]	[503, 503]	[12503, 14503]
41	[2019-02-21 16:00:00, 2019-02-21 16:39:00, 201...]	[2019-02-22 08:16:24, 2019-02-22 08:17:25, 201...]	[Makas, Hakan Ateş'ten Denizcilerimize Mesajla...]	[16000, 16000, 12000, 12000, 11000, 16000, 110...]	[100, 100, 100, 100, 100, 97, 100, 1...]	[2018-05-16 19:43:48, 2018-05-16 19:43:48, 201...]	[Friday, Friday, Friday, Friday, Monda...]	[8, 8, 11, 14, 15, 14, 15, 15, 15, 15, 9, 10, ...]	[500, 500, 500, 500, 500, 100, 400, 100, 100, ...]	[508, 508, 511, 514, 515, 114, 415, 115, 115, ...]	[16508, 16508, 12511, 12514, 11515, 16114, 114...]
47	[2020-10-07 12:45:00, 2020-10-07 12:45:00, 202...]	[2020-10-07 14:27:21, 2020-10-07 14:27:22, 202...]	[Ekipleri Uzaktan Etkili Yönetmek, Ekipleri Uz...]	[5000, 5000, 5000, 1000]	[100, 100, 100, 85]	[2018-06-21 08:01:03, 2018-06-21 08:01:03, 201...]	[Wednesday, Wednesday, Wednesday, Wednesday]	[14, 14, 14, 14]	[300, 300, 300, 300]	[314, 314, 314, 314]	[5314, 5314, 5314, 1314]
53	[2018-11-01 08:30:57, 2018-11-08 08:30:00, 201...]	[2018-11-17 13:04:53, 2018-11-17 13:06:15, 201...]	[Çalışan Bağlılığı 3.0: Kişiyi Özel "Nudge (Dü...]	[11000, 14000, 14000]	[100, 100, 100]	[2018-06-27 07:53:16, 2018-06-27 07:53:16, 201...]	[Saturday, Saturday, Saturday]	[13, 13, 13]	[600, 600, 600]	[613, 613, 613]	[11613, 14613, 14613]

- c. Then, we added the corresponding content type of each hour time on each day of the week. We first made changes to the categorical variable. The numbers of content type are stored as integers from 1 to 17. To make it easy to parse the day, hour, and content type, we multiply each integer by 100. We added hr_num,

	Activation Date	Activity Date	Name	Content Type	Action	User Created At	Activity Day	hour	day_num	total_num	Content_Type_total	count_total	every_day_fr		
User Id															
21	[2018-10-31 13:32:19, 2018-11-01 08:30:57, 201...	[2018-10-31 14:08:53, 2018-11-01 08:33:27, 201...	[Çalışan Bağlılığı 3.0: Kişiye Özel "Nudge (Dü...	[11000, 11000, 11000, 12000, 14000, 14000, 140...	[100, 100, 100, 100, 100, 100, 100, 100, ...	[2018-05-14 02:49:06, 2018-05-14 02:49:06, 201...	[Wednesday, Thursday, Thursday, Wedn...	[11, 9, 15, 15, 15, 15, 16, 16, 8]	[300, 400, 400, 400, 300, 400, 300, 300, 300, ...	[314, 408, 408, 408, 311, 409, 315, 315, 315, ...]	[14116, 14311, 14409, 14315, 11408, 12208, 113...	[(315, 4), (408, 3), (116, 2), (314, 1), (311,...]	[[[(116, 2)], [(20 1)], [(315, 4 [(408, 3)		
	38	[2019-10-03 08:00:00, 2019-10-03 09:00:00]	[2019-10-04 10:12:57, 2019-10-04 03:13:10]	[demo filip, safety]	[12000, 14000]	[100, 100]	[2018-05-14 17:59:52, 2018-05-14 17:59:52]	[Friday, Friday]	[3, 3]	[500, 500]	[503, 503]	[14503, 12503]	[(503, 2)]	[[. [. [. [. [. (50 2)], [. [.]]	
		41	[2019-02-21 16:00:00, 2019-02-21 16:39:00, 201...	[2019-02-22 08:16:24, 2019-02-22 08:17:25, 201...	[Makas, Hakan Ateş'ten Denizcilerimize Mesajla...	[16000, 16000, 12000, 12000, 11000, 16000, 110...	[100, 100, 100, 100, 100, 97, 100, 1...	[2018-05-16 19:43:48, 2018-05-16 19:43:48, 201...	[Friday, Friday, Friday, Friday, Monda...	[8, 8, 11, 14, 15, 14, 15, 15, 15, 15, 9, 10, ...	[500, 500, 500, 500, 500, 100, 400, 100, 100, ...	[508, 508, 511, 514, 505, 514, 515, 114, 415, 115, 115, ...]	[8211, 12310, 12312, 16410, 12315, 16413, 7712...	[(410, 63), (309, 17), (411, 14), (111, 14), (...]	[[[(111, 14), (11 11)], [(215, 11 (210, 9)]

In the above screenshot, in order to meet the demands of the client for the customer impression, we used frequency and average to calculate the completion rate at same time. We first gathered all the frequency for each user at their different optimal notification sending time. To test the closeness of frequency, we use the average function. For each day, we find the average frequency of each hour. If the frequency is larger than the average, we will remain the hour and frequency; if not, we will ignore the hour time. As you can see above, the column “every_day_freq” is the result that is over the average frequency. This helps us determine which content type is the most popular and acceptable by the users overall, which can be used to send notifications for our new users (secondary goal of our project). At the same time, we can also check the frequency of each user and see which content type they like the best which can be used to increase the completion rate of each notification.

3. Developing stage 2

- a. Next, our team wanted to combine content type with day and hour as a list. We used list(set()) to get a list of content_type_total with unique elements for each user. We also defined a new function nth_digit to return the nth digit of an integer. By using this function, we could easily find out the day by pointing to the 3rd digit. By for-loops and nth_digit, we could combine day and hour with the corresponding content type. The result is shown below:

User Id	Mon	Tue	Wed	Thu	Fri	Sat	Sun
21	[[16, 2, 14]]	[[8, 1, 12]]	[[15, 4, 14, 11, 12]]	[[8, 3, 11, 12]]	[nan]	[nan]	[nan]
38	[nan]	[nan]	[nan]	[nan]	[[3, 2, 14, 12]]	[nan]	[nan]
41	[[11, 14, 1, 16, 14, 12], [15, 11, 1, 7]]	[[15, 10, 14, 12], [10, 9, 16, 14, 12]]	[[9, 17, 16], [10, 13, 12, 4, 16], [11, 10, 11...]]	[[10, 63, 16, 14, 12, 7, 4], [11, 14, 12, 7], ...]]	[[9, 12, 16, 8]]	[[17, 6, 16, 12]]	[[12, 6, 7]]
47	[nan]	[nan]	[[14, 4, 5, 1]]	[nan]	[nan]	[nan]	[nan]
53	[nan]	[nan]	[nan]	[nan]	[nan]	[[13, 3, 14, 11]]	[nan]
...
49651	[nan]	[nan]	[nan]	[nan]	[nan]	[[14, 14, 14, 12, 16]]	[nan]
49652	[nan]	[nan]	[nan]	[nan]	[nan]	[[14, 21, 14, 12, 16]]	[nan]

- b. As you can see above, in the column representing Monday, “Mon”, the result follows the format as follows: user_id: [[time1, frequency1, corresponding content_type], [[time2, frequency2, corresponding content_type] ...]. Take user id 21 as an example. On Monday, that user finished 2 times at 16:00 (4 pm) with the content type of 14. On Wednesday, that user finished 4 times at 15 with content types of 14, 11, and 12. So that our client could easily recognize the data that we produced and find the information that they need. Then, we stored all the optimal notification sending time in the csv documents. In the csv file, the

columns include: user_id and each day of the week, so that the client can use the csv file and can be imputed in SQL or other software and easily used.

4. Developing stage 3

- a. For the users whose completion rate is below 85, we also gathered a dataframe just in case our client may need this file for future improvement. So we completely separate the completion rate from 85 to 100 and users below 85 and list them in a new table.

User Id	Mon	Tue	Wed	Thu	Fri	Sat	Sun
105	[nan]	[nan]	[nan]	[nan]	[23, 1]	[nan]	[nan]
510	[nan]	[nan]	[22, 1]	[nan]	[nan]	[nan]	[nan]
796	[nan]	[nan]	[15, 1]	[nan]	[nan]	[nan]	[nan]
898	[nan]	[nan]	[[16, 1], [17, 1], [18, 1]]	[nan]	[nan]	[nan]	[nan]
979	[nan]	[nan]	[nan]	[nan]	[nan]	[23, 1]	[nan]
1012	[nan]	[nan]	[18, 1]	[nan]	[nan]	[nan]	[nan]

In the screenshot, the format is the same as the before except the completion rate is below 85. We are still not so sure what format that the client may want. So we still need to change our code in the future to meet the client's demands.

Lessons Learned and Insights:

During this iteration we have a few improvements. Followed by iteration 2, we projected the different content type to number 1 to 17, instead of one-hot coder so that the client can easily use and understand. And we times 1000 to "content type" column, 100 to "activity date" column and add with "hour" column together to save the algorithm time and easily to calculate. Obviously, compared with string, numbers can be easily calculated and imported into different softwares.

Secondly, we used an average frequency function to classify the customer impression. Instead of storing all the time that is suitable for the optimal notification sending time, we need to see the customer impression and then to decide. So we added the average frequency on base of iteration 2. When the frequency is below the frequency average we will just ignore that time and focus on the time that has a better impression.

Pitfalls and Future steps:

Also, this iteration is not as smooth as we write in the project report. Our team also met the pitfalls and solved it as best as we can. When the client had the request of considering the customer impression and adding content type. We had several plans to do that and not sure what

format the client really wanted since we also need to add content type in the final result. As we mentioned above, we use a transformation from string to number type to solve the content type and customer impression problem. As to the format of final result. We communicated several times and we decided to use the format as user_id : [[Day1, time1, frequency1, corresponding content_type], [Day2, time2, frequency2, corresponding content_type] ...].

For the future step, we may want to use the data results we get so far to train the model of the algorithm. So that we can predict the optimal notification sending time for the new users. In addition, the client can use our data analysis and training model in the future, so that the client doesn't need to spend more money on the data analysis processing. For the different algorithms, we may try linear regression, logistic regression, and other algorithms and adjust hyper-parameters to find the best-performed algorithm.

5. Evaluations & Questions for our client

By now, we have found out the prediction for users with action above 85 inclusively with corresponding content types.

For users with action below 85, though we have created a dataframe and deleted all the duplicated user id from action above 85, we are not sure how to reflect this part (in which format) to the client.

6. Reference

Marketing. (2019, July 15). The Science Behind Send Time Optimization - The Robly Blog.
Retrieved November 27, 2020, from
<https://blog.robly.com/2019/07/16/the-science-behind-send-time-optimization/>