Send Time Recommendation

Leandro Corrêa Gonçalves

I. Definition

Project Overview

Digital Marketing is an industry that is constantly looking for ways to communicate with customers in a personalized way in order to keep them engaged with the Company. With this engagement, it is possible to increase revenue by keep people frequently spending on its products/services or even making them spend more along their relationship time with the company.

Email is one of the most used communication channel in Digital Marketing Campaigns, and CRM systems usually provide many features to help companies reaching their customers. So for those campaigns, enhancing email open rate means increase the audience exposed to your ads which might influence more people to buy.

Therefore this project proposes to use Machine Learning approaches to build a recommender that is able to predict optimal email delivery times using customers preferences extracted from their past activities data, so the open rate can be increased and consequently affect others metrics such as Revenue, Churn Rate, Repurchase Rate, among others related.

Problem Statement

What is the best time to send emails to my customers in order to increase open rate?

People tend to open recent emails first as they are sorted in chronological order, so as new incoming email arrives, the last ones are moved to the end or even another page of their inboxes hence it can be missed or deleted without being opened. When we talk about Digital Marketing, email is a very common communication channel and open rate is frequently used to measure how much of the desired audience have been reached by company's campaigns, so higher number of open rate means the possibility to influence more customers and drive them to a future purchase. For this reason, the time that an email is sent is an important piece of campaigns notifications and finding an optimal

time to dispatch those emails can increase the chance of customers to open, placing them on the top of their inboxes.

Common approaches to optimizing emails delivery time like running A/B Tests and check the posterior results from control and optimized groups or using most frequent send times to suggest the best time could work, however, Machine Learning algorithms provide better and suitable solution based on user's past activities data and similar users behavior in addition to explore new optimal time slots.

Solution Statement

In order to solve this problem it's necessary to define how quantify user's preferences according to their open times recorded. This can be achieved by calculating the proportion of opened emails over the total emails received in each time slot (e.g. 3 pm or 10 am), so it will be possible to find which are the most common time(s) the users open their emails. After that, the users can be grouped with similar users using clustering algorithms, having these proportions as its features. Once it has the most common (and optimal) time for each of customer's group (clusters), it will be possible to recommend optimal send times.

Another point that deserves attention is the need of exploring new time slots because it might be useful for tracking changes in user's behaviors and also necessary to maintain and increase the model in a long term perspective. To do so, we need the recommender to explore additional time slots instead of always use the optimal group's time, performing random choices giving the probabilities of choosing a time slot for a given cluster.

Moreover it will be necessary to weight these probabilities by their importance for each customer.

For example:

For a given customer the most probable times would be:

1. Group's optimal time
2. **Next most probable cluster** time (in a soft-clustering approach)
3. Time of next most probable cluster of his **similar users**.

Finally we can use the probability of the remaining time slots of the user's cluster ensuring time slots exploitation and exploration, that is, using the most optimal time but often trying to figure new time slots to track any possible change in user's behavior.

Metrics

The best metric to evaluate the solution's performance is the KPI attached to this problem which is Email Open Rate. The true value only can be measured using A/B tests with an optimized group with time recommendation and another control group with no optimization so it can confirm the solution's feasibility and/or significant improvements.

In development stage the data will be split into two groups: Train and Test sets with recent periods assigned to test, so the model should be able to suggest open times in future unseen data.

As the final model outcomes would be classes that represents hour-ranges, the overall right predicted classes will be evaluated using **Micro Average F1-Score**. It's a simple but robust metric as it balances Precision and Recall metrics, prevent bias like always predict the most frequent class and works for multiclass prediction with imbalance class issues, in cases like 04-05 a.m with very few openings while most of the data concentrated at dinner and rush times.

F1-Score is calculated as follows:

2* (Precision * Recall / (Precision + Recall))

Precision: Number of correct positive predictions called True Positives / Total of cases predicted as positive

Recall: True Positives / Number of cases that are actually positive (and not necessary has been predicted as positive)

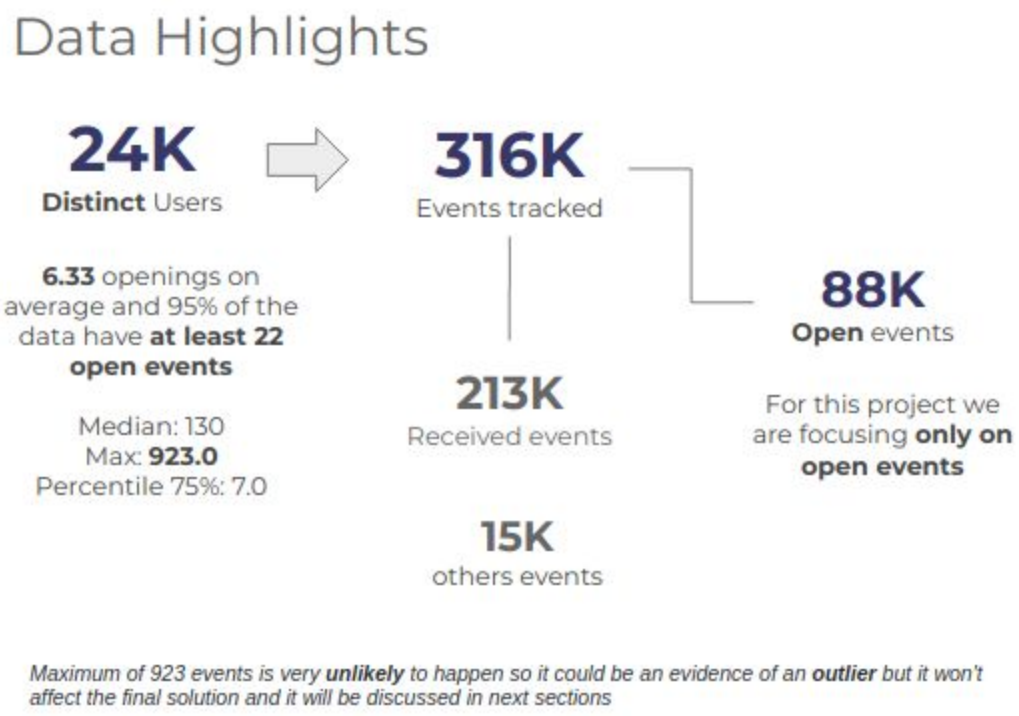
II. Analysis

Data Exploration

The dataset that will be used in this project is a set of events related to emails dispatched and their respective returns. These events are generated by each interaction of the users and collected by a CRM system, so by the time a person received and/or open an email, it generates an event with the respective action. This data has been used in a Data Scientist challenge from a Brazilian Company named Dito and it is anonymized. The dataset columns are describe below.

- **id**: user identify
- **timestamp**: event date and time
- **email_id**: email sent identify
- **action**: The event action
    - ➔ **received**: indicates that user received the email
    - ➔ **open**: user opened the email
    - ➔ **click**: user clicked on some link that was in email's body
    - ➔ **spamreport**: User reported spam.
    - ➔ **unsubscribe**: User unsubscribe for this email

Note: *There might be more than one event for the same user_id and email_id, as the same user can received, open, click, report spam and unsubscribe one single email.*

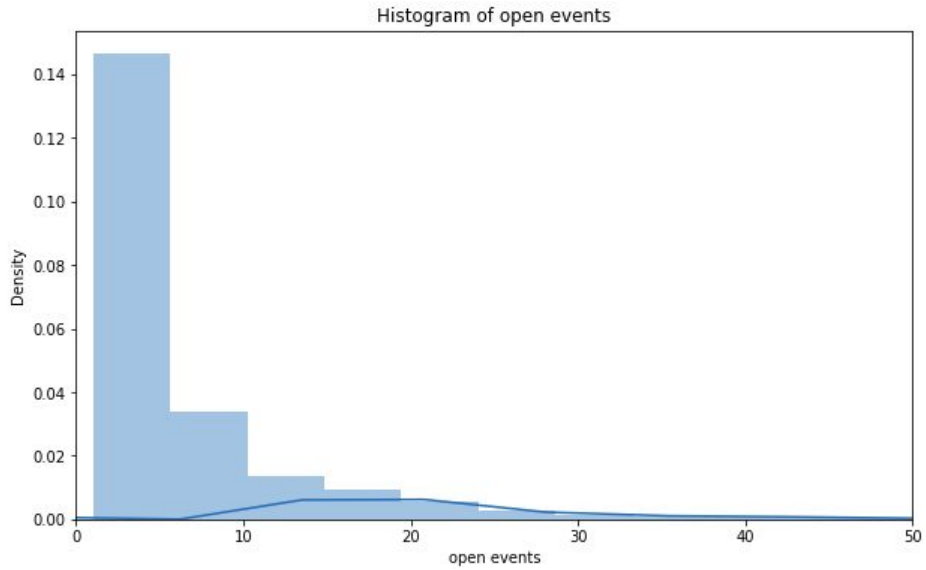


*Maximum of 923 events is very **unlikely** to happen so it could be an evidence of an **outlier** but it won't affect the final solution and it will be discussed in next sections*

Exploratory Visualization

In exploratory phase some business questions has been analyzed to support some previously formulated hypothesis in addition to explore the data itself and extract important details that could be used on the final solution. Therefore, the next topics would be organized with the TOP 3 questions that has driven most of analysis and visualizations made.

1 - How does this data is distributed?



Mean	Std	Min	Perc 25	Median	Perc 75	Perc 95	Max
6.33	12.25	1.0	1.0	3.0	7.0	22.0	923

The open events data follow a right skewed distribution with most of events concentrated up to 10 openings with 75% of people with less than 7 openings and 95% with 22 events, so it means that having customers with more than 22 openings is unlikely to happen on this data.

Customers have 3 openings on average (median) and the maximum of open events for the same customer was 923 in the selected period (January to November 2018). As it is extremely higher than the rest of the data, it could be the case when sellers have a “default” email address and assign it to every customer on offline purchases in physical

stores or at least for those who do not provide their personal information when the sellers have to register the purchase.

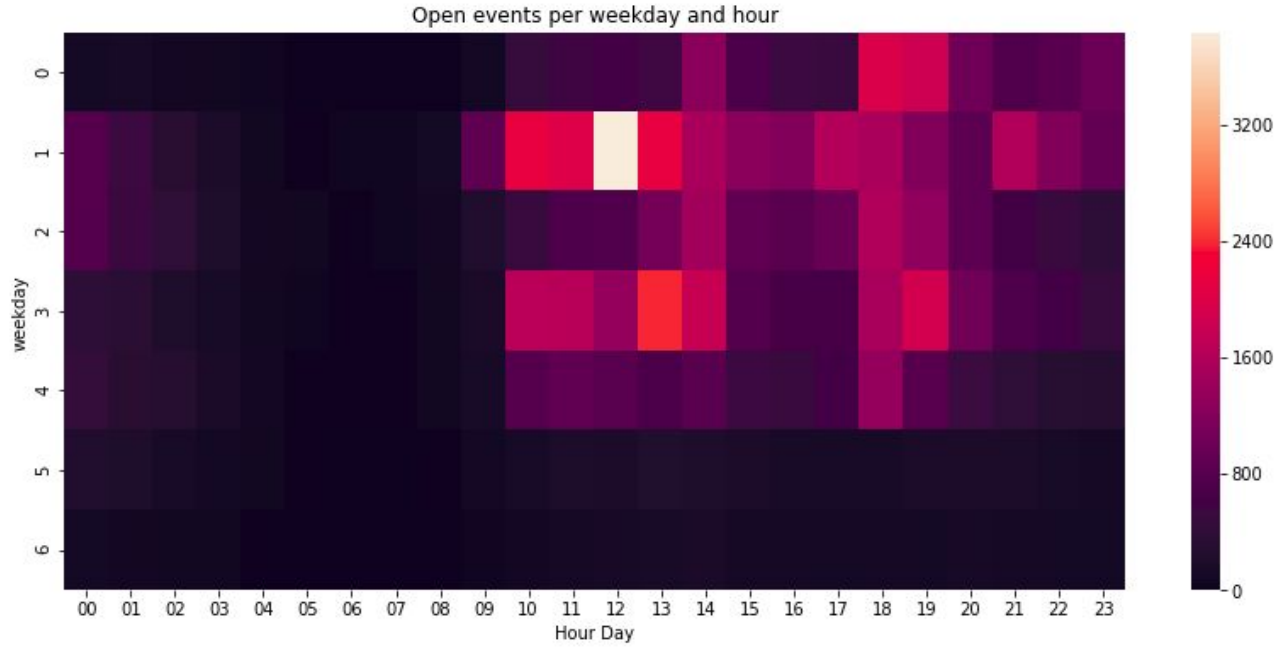
Outliers and/or atypical cases will not affect the final results as the recommender is designed to work with proportions so the data will be normalized.

Further discussion on this matter will be present on solution design topic.

2 - Does the weekday influence open behavior?

It's expected to have some differences depending on the weekday as people's routines often change especially during the weekend and the recommender should be able to extract those differences.

These differences can be easily perceived by the following visualization:



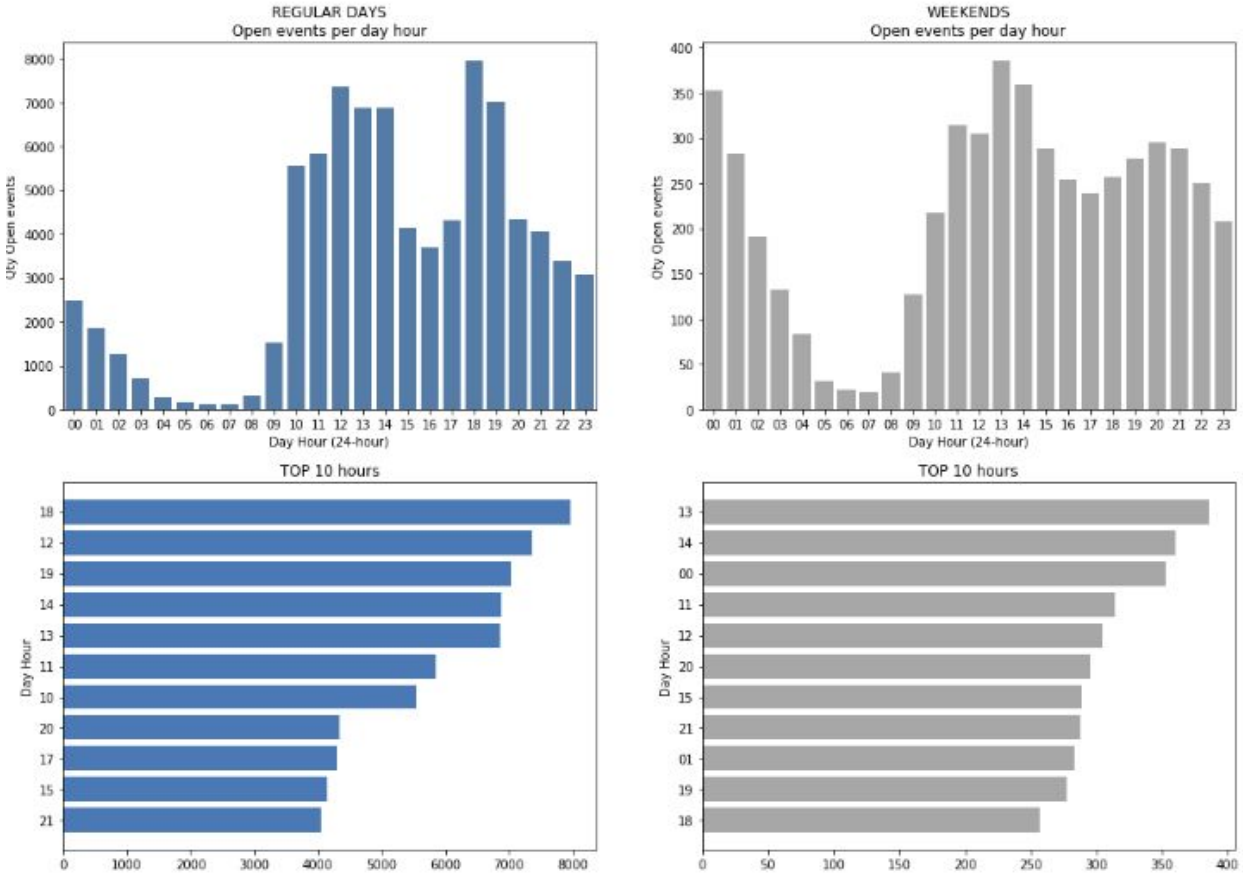
The X-axis represents the day hours and the Y-axis the week days (0 = Monday ... 6 = Sunday). The color heat is the sum of open events meaning that the lighter the color the higher is the number of open events.

It is possible to notice a concentration of open events On Mondays to Fridays (day 0 to 4) with a clear contrast in hours 10 to 13 (in 24-hour format) especially on Tuesdays and Thursdays. So we can see a big difference on user's behavior on regular days when compared to weekends

By the graphs below we can see a higher concentration in morning hours on the weekends and differences on TOP 10 hours, for example while 18 is the most frequent hour for regular days, it is only the tenth most frequent on the weekends.

Another interesting point is the presence of hour 00 on weekend's TOP 10 as it's not even present on regular days

Therefore those differences make it clear that **weekday is an important feature to consider on send time recommendation.**

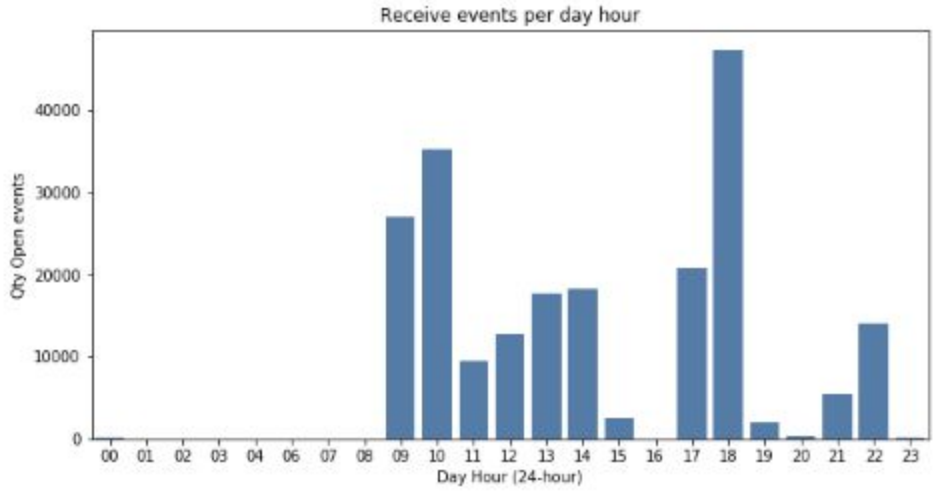


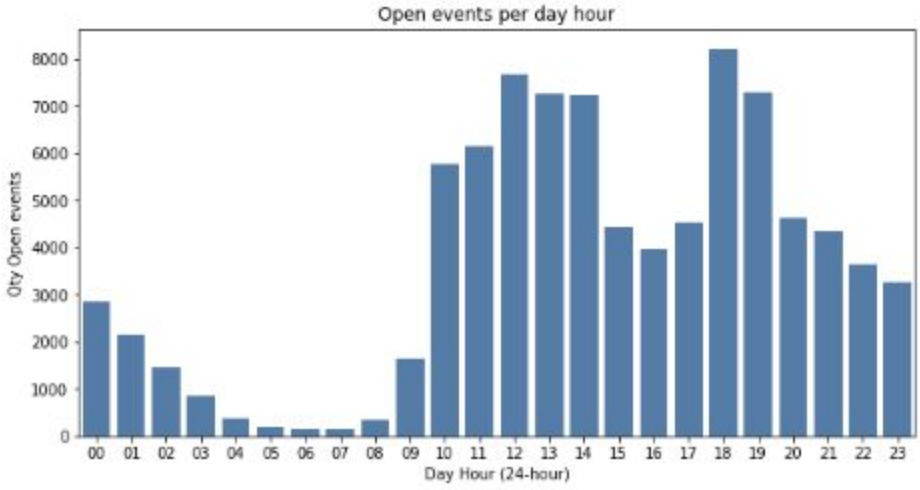
3 - Is the open time directly affected by receive time?

This is a question that could invalidate the send time recommender because if the users are more likely to open their emails at the moment they receive them, a time recommender would not be necessary.

With this, **receive** events will be analyzed to support the hypothesis of having more open events concentrated in specific times is due to the amount of emails received at this same time.

First let's compare how open and receive events are distributed along the day hours and see that in general, regular days and weekends concentrations are not following the same shape.



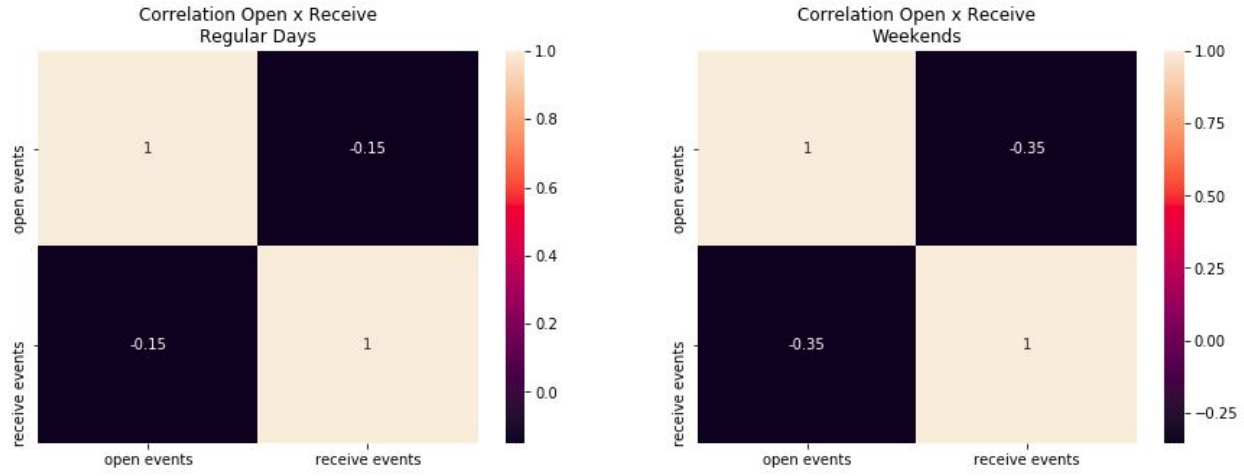


Before any confirmation that open events might be related to people's behavior instead of receive time, two more tests might support this hypothesis: correlation and average time between open and receive events.

*if the hypothesis of people opening their e-mails at the moment they receive is true, correlation coefficient should be **high** and average time should be **low** as the lower the*

time between open and receive time stronger the evidence in favor of receive time influencing open time.

Correlation:



There is almost no linear correlation between Open and Receive events according to the correlation matrices above (-0.15 of correlation index for regular days and -0.35 for weekends)

Average time Open x Receive:

The difference between receive and open time for the same customer and email has been calculated to obtain the average time and the descriptive statistics are reported below:

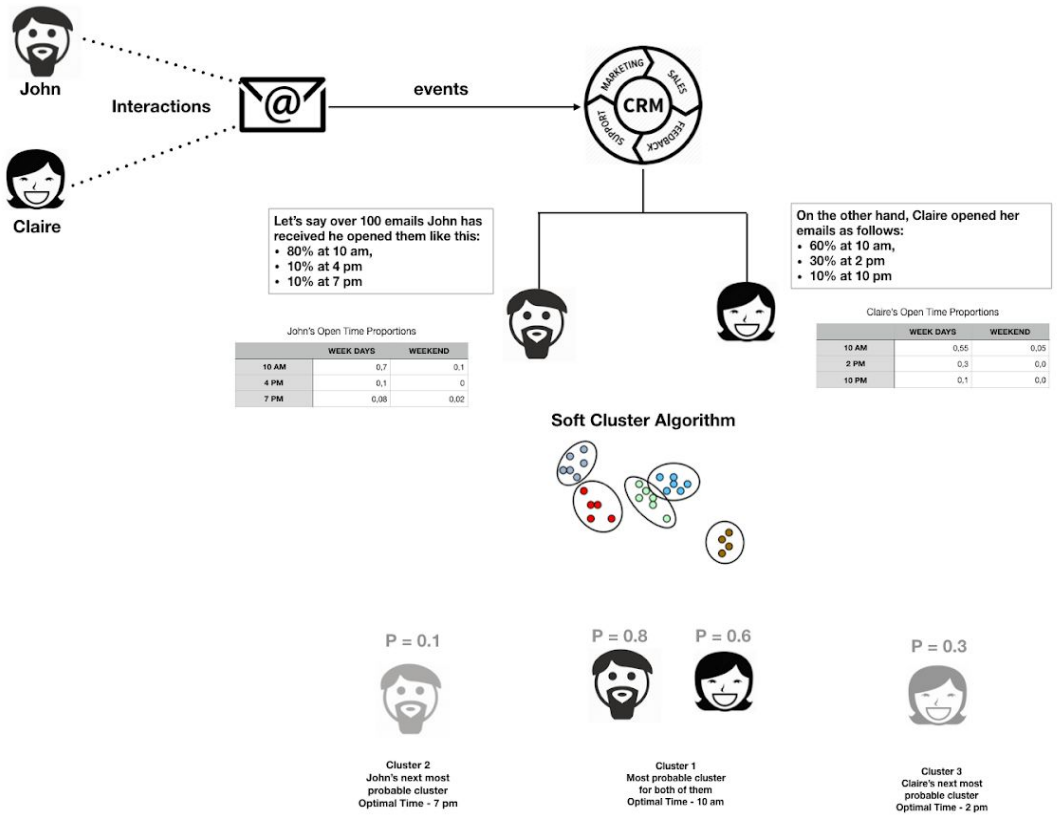
count	88380
mean	2 days 22:15:10.205091
std	11 days 09:47:38.243731
min	0 days 00:00:01
25%	0 days 00:58:26
50%	0 days 04:20:32.500000
75%	1 days 00:42:01.250000
max	286 days 18:06:33

Over more than 88k identified cases with open and receive events for the same email and customer, the median of **4 hours and 20 minutes** reinforces there is no clear evidence that they are directly related, otherwise the median will be lower meaning that open and receive time would be closer.

With the presented analysis it was possible to check some business questions before moving forward with recommender implementations and also know the data better. Each of those questions helped to guide the analysis and extract important characteristics.

Algorithms and Techniques

This solution has been framed as a recommendation problem so for this reason is essential to group similar people together according to their similarities. For this kind of problem clustering algorithms can be used to extract user's similarities and create distinct groups, but before the algorithm's discussions, the image below illustrates how the solution is designed and helps to justify the chosen technique.



It is all about recommend an hour range by similar people behavior and their own behaviors so the optimal time(s) to send an email to John would be:

1. The optimal time from John's **most** probable cluster.
2. The optimal time from John's **next most probable** cluster.
3. The optimal time from the next probable cluster of John's **similar** users.
4. The remaining probabilities of each time slot from John's most probable cluster.

How do we know that someone is similar to another with data?

If a given person usually opens emails in the morning and lunchtime, a similar user would be someone with the same preferences of times, so to identify similarities and group people the proportions of open emails along the time ranges* will be used

Therefore if 80% of John's emails are opened at 10 A.M in regular days, and the remaining 20% at 4 p.m and 7 p.m, the goal is to find people with similar habits and assign them in the same John's group.

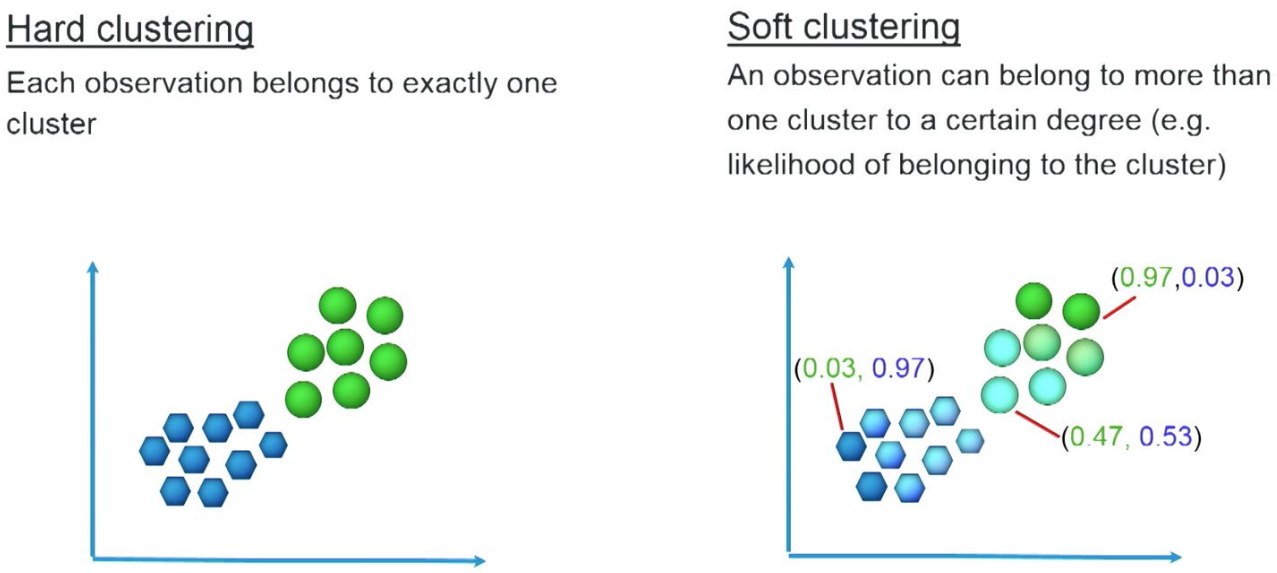
**Day ranges are time slots created to divide the day hours in two hour ranges*

Clustering Algorithm

Clustering is basically group similar people within distinct clusters so we can understand and guide our actions according to people's differences and similarities. In the context of this project, similar people would be the ones that usually have the same behavior related to the time of open emails.

Hard Clustering x Soft Clustering

There are two kinds of clustering approaches: Hard and Soft. The image below demonstrates the main difference between them.



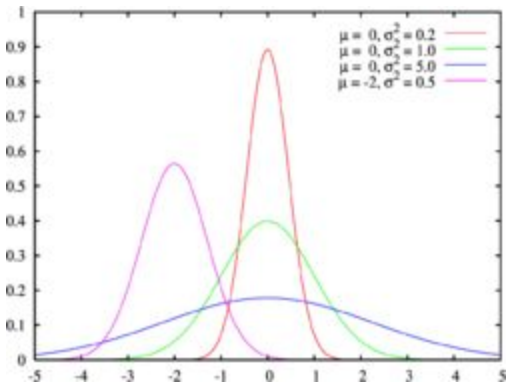
Which one should be used for this project?

As soft clustering algorithms return the probabilities of a given customer belongs to any of the distinct groups, it has an important role on the proposed solution as it enables to capture different behaviors for the same person as routines usually change due external factors like vacation, night classes, etc.

The probabilities outcomes will be used to **explore new time slots** in addition to the most probable cluster of each customer, so it will be able to suggest additional times from others clusters that people have the chance(probability > 0) to belong.

Gaussian Mixture Model (GMM)

The most common definition of this algorithm is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing *k-means clustering* (which is a hard cluster algorithm) to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.



It is like we assume that we have different density distributions in the same data and it tries to capture those differences and assign probabilities to each customer of being on the density distributions found.

Further details about GMM parameters and functionalities on this link.

With that said, GMM algorithm will be used on this solution to cluster people according to their open behavior. The input dataset for the algorithm will be customers proportions in each hour range so the algorithm can group them using a metric of distance like euclidean distance that calculates the distance between two points in a plane.

	08-09	09-10	10-11	12-13	13-14	14-15	...	23-24
Customer 1	0,3	0	0,6	0	0	0,1		0
Customer 2	0	0,4	0	0,5	0	0,1		0
. . .								
Customer N	0	0,1	0	0,8	0	0	0	0,1

Benchmark

No official paper or published work that could be used as a benchmark have been found by the time of this research. Most Companies with this kind of feature implemented does not provide detailed information about how exactly their algorithms work and its parameters. However a simple but effective method defined as benchmark model will be **using the most frequent open time of each customer to suggest their optimal send time**.

The Company Emarsys has described part of their solution on this link that can be used as a reference, despite it is a different approach. Another Company named CleverTap seems to have something similar to the benchmark model defined called Best Time for Batch Campaigns using the most frequent hour range to suggest the optimal time, letting the end user to choose which event (apart from open events) he wants to be considered in the final recommendations.

The main goal of this solution is to increase Open Rate metric of Digital Marketing campaigns, but the project's success criteria is to predict an hour range that people would open emails, as we believe that the time an email arrives **is one of the factors** that could drive someone to open it. So for the cases that time is a factor that influences the openings, optimal times suggested could lead these people to open more emails. As it is still unknown how much the time influences the open rate, there is no improvement target defined by Dito before running A/B tests and explore the margin of improvements, however it's expected **some** increase in the initial tests.

To conclude the discussion about benchmark, there is an interesting research about references of open rate for each kind of Industry from Mailchimp Company, which is a widely used Company to dispatch emails campaigns. Those numbers will be used to guide our improvement analysis. By looking on Dito's customers set of campaigns, open rate hardly hit 18% and we expect to increase this up to at least 20% on average.

Tables of Average Open Rate by Industry (March 2018)

Source: https://mailchimp.com/resources/email-marketing-benchmarks

Industry	Open Rate
All non-labeled accounts	21.09%
Agriculture and Food Services	23.12%
Architecture and Construction	23.13%
Arts and Artists	26.03%
Beauty and Personal Care	17.01%
Business and Finance	20.47%
Computers and Electronics	19.39%
Construction	21.01%
Consulting	18.96%
Creative Services/Agency	21.59%
Daily Deals/E-Coupons	14.92%
ecommerce	15.66%
Education and Training	21.80%
Entertainment and Events	20.41%
Gambling	18.47%
Games	19.71%
Government	26.52%
Health and Fitness	20.06%
Hobbies	27.35%
Home and Garden	22.21%
Insurance	20.99%

Industry	Open Rate
Legal	21.14%
Manufacturing	20.51%
Marketing and Advertising	16.48%
Media and Publishing	21.92%
Medical, Dental, and Healthcare	21.09%
Mobile	18.41%
Music and Musicians	21.80%
Non-Profit	24.11%
Pharmaceuticals	18.95%
Photo and Video	22.99%
Politics	22.30%
Professional Services	20.77%
Public Relations	20.21%
Real Estate	19.67%
Recruitment and Staffing	19.33%
Religion	25.33%
Restaurant	20.26%
Retail	19.36%
Social Networks and Online Communities	21.13%
Software and Web App	19.81%
Sports	23.77%
Telecommunications	20.27%
Travel and Transportation	20.03%

III. Methodology

Data Preprocessing

In this stage the dataset had to be prepared to extract the input features to GMM cluster algorithm. The initial dataset described on Data Exploration section, has a set of events of user's interactions with notification (email in the scope of this project) with the timestamp, the action description and a customer identifier. So the process to transform this data in features to the algorithm had three main steps detailed below.

Step 01 - Create additional columns:

Some additional columns have been included in the dataset so it could be transformed in the final desired format.

- **flg_open:** A flag field to mark the record whether is an open event (1) or not (0)
- **flg_received:** A flag field to mark the record whether is an receive event (1) or not (0)
- **event_time:** The time extracted for the timestamp field in the format HH:MM:SS. This column is necessary to extract the hour number.
- **event_date:** The date extracted from timestamp field in the format YYYY-MM-DD. This column is necessary to extract the weekday.
- **event_hour:** The hour number extract from event-date field in the format HH. This is necessary to hour range column.
- **event_month:** The month when the event happened extract from timestamp field in the format MM.
- **weekday:** The number of weekday extracted from event_date field. This column is necessary to calculate the flg_weekend.
- **flg_weekend:** A flag field to mark the record as weekend (1) or regular day (0).
- **hour_range:** The day hours extracted from event_hour field divided into slots of two hours.
- **day_type**: Group of days: 0 for even days, 1 for odd days and 2 for weekend

So the data was transformed from this format:

id	timestamp	email_id	action
4591b11ba8cca67079c1a43be2992a8f89fce422	2018-05-25 14:59:02 UTC	3498910	open
2bbdb4cff0fe8cc3cb6c1757291e31806ecefa47	2018-06-25 17:47:23 UTC	3498910	open
0f467135eabd4e385f9c2dcd3f00a9f2a04c0115	2018-06-27 12:32:36 UTC	3498910	open
6f17b7dc6f220c09fba4d8fbc2491317eece2ae8	2018-07-02 22:47:28 UTC	3498910	open
a645dd2ac5c5e000c8b5c7739b3a54435fb313cc	2018-01-24 19:21:49 UTC	3498910	open

To this format:

id	timestamp	email_id	action	flg_open	flg_received	weekday	flg_weekend	event_time	event_hour	hour_range	event_month	event_date
;be2992a8f89fce422	2018-05-25 14:59:02 UTC	3498910	open	1	0	4	0	14:59:02	14	14-15	05	2018-05-25
7291e31806ecefa47	2018-06-25 17:47:23 UTC	3498910	open	1	0	0	0	17:47:23	17	16-17	06	2018-06-25
:d3f00a9f2a04c0115	2018-06-27 12:32:36 UTC	3498910	open	1	0	2	0	12:32:36	12	12-13	06	2018-06-27
)c2491317eece2ae8	2018-07-02 22:47:28 UTC	3498910	open	1	0	0	0	22:47:28	22	22-23	07	2018-07-02
39b3a54435fb313cc	2018-01-24 19:21:49 UTC	3498910	open	1	0	2	0	19:21:49	19	18-19	01	2018-01-24

Step 02 - Group data by customer_id and hour_range:

After additional columns included, some aggregation was performed to have the number of emails opened for each customer and hour_range. Besides the total of opened emails for each customer was also included to be used in the final calculations.

The input data was looking like the previous transformed data and the final result of this step is like this:

		flg_open	flg_open_total
id	hour_range		
00009d30d2f6cdca3b306b6a0fe58a32e51dabdc	12-13	1	2
	18-19	1	2
00074efe132b0b36c90ef600386ca2f863bc5e49	14-15	2	2
0009b0b413429e75bf81c8989d7c6201f2d86180	10-11	1	6
	12-13	2	6

- Each row is a customer id,
- the hour range column represents the time slots,
- flg_open is the number of emails opened within that hour range and
- flg_open_total is the sum of all emails opened of the customer in the row

Step 03 - Calculate proportions

In this final preprocessing step, the data was pivoted to have each hour range and proportions as the cluster algorithm inputs so we can use the proportions of openings in each time slot as the features to calculate similarities, that means when two customers have similar proportions (number of emails opened at an hour range / total of openings) in a specific time slot, they would be grouped together.

hour_range id	00-01	02-03	04-05	06-07	08-09	10-11	12-13	14-15	16-17	18-19	20-21	22-23
00009d30d2f6cdca3b306b6a0fe58a32e51dabdc	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.50	0.00	0.00
00074efe132b0b36c90ef600386ca2f863bc5e49	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
0009b0b413429e75bf81c8989d7c6201f2d86180	0.00	0.00	0.00	0.00	0.00	0.17	0.33	0.00	0.00	0.50	0.00	0.00
000e522c0071795551e9ebc958ea80be87d64ec3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.33	0.00	0.00
001a8819c8fc79e60c8a580f78a6502b45d9206c	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00

Having data in proportions format we exclude the effects that outliers could bring to any machine learning model as we can work with every customer in the same unit (proportions with decimal numbers from 0 to 1). Therefore it is possible to compare (and group by similarity) a customer that had opened 1K *versus* another customer that opened only 5 for example, as long as we have their proportions calculated and distributed along the hour ranges.

Weekend vs Regular Days

As we have seen in Data Exploration, weekends and regular days follow different openings behavior, so three different cluster classifiers will be trained, one for each day type, instead of using flg_weekend as a feature to the same classifier. In this way the cluster numbers can be reduced to each classifier (as the dimensionality also is reduced hence the complexity and possible combinations), and it will be able to provide better suggestions depending on the weekday that an email will be sent. So in the final solution, weekday will be a parameter required to the recommender algorithm.

Implementation

The previous discussions have illustrated the dataset final format, the solution design and also the reason GMM clustering algorithm is the core of the recommendation process proposed. The solution's pipeline steps will be discussed and explained in next sections.

During the recommendation process, it needs to predict two different things:

1. A **cluster** to a specific customer according to his/her data, in other words which group a "new" customer would fit better.
2. The **hour-range** to send an email for a specific customer and date.

For the first case, three classifiers need to be built: One for weekend recommendations, one for odd days and another one for even days. This is necessary as we are trying to capture different behaviours along the week, for example routine changes like night classes and courses often happen either in odds days like Tuesdays and Thursdays or even days (Monday and Wednesday and Fridays). The hour range predictions will use these classifier's probabilities predictions to select which hour range will be recommended given the input data provided.

These are the steps needed in recommendation process followed by a practical example to make it clear how the recommender works.

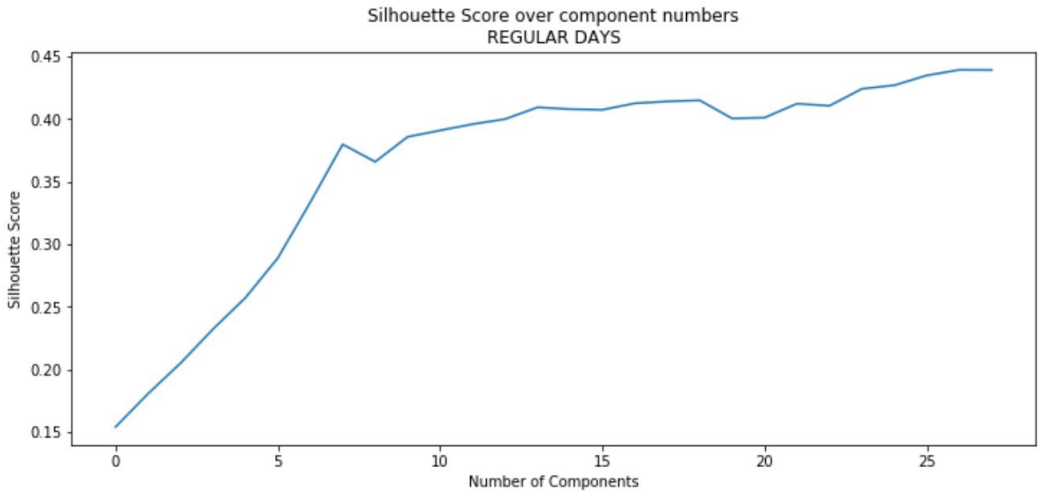
01 - Training phase:

1.1 - Data split into train and test sets to the clustering algorithm: Data needs to be split in two datasets so we can evaluate cluster's predictions performance: One for training with a ratio of 80% of the data and another for tests with the remaining 20%. It can be achieved by randomly selecting customers ids and keep them in the test list.

1.2 - Build and train the GMM classifiers (Regular days and weekend classifiers).

There are some techniques to guide the choice of cluster numbers parameter. These techniques measure the algorithm's performance using N different clusters and show which number hit the best performance based on the chosen method.

The method used was the Silhouette Score which is a metric score with values from -1 to 1, being -1 the worse value and 1 the best for a certain number.



Here is an example of how the Silhouette Score increases along with the number of clusters

02 - Recommendation phase:

2.1 - Predict the probabilities of a given new customers in each cluster.

After the classifier training, we can see how it's performing on cluster's predictions by comparing a sample of customers and their similarities. To consider that the model is good at assigning customers group, it's expected to have similar people (people with similar proportions on the same hour-ranges) within the same group and different from other people's groups.

Sample of customers from Cluster 0

```
df_reg_train[df_reg_train['cluster'] == 0][0:2]
```

hour_range id	00-01	02-03	04-05	06-07	08-09	10-11	12-13	14-15	16-17	18-19	20-21	22-23	cluster
00a3d6aa039587031a6c88a9d3de389bd54865ab	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0
01323bb09a7cd9c372b8bcffcbd278de3c6643bb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0

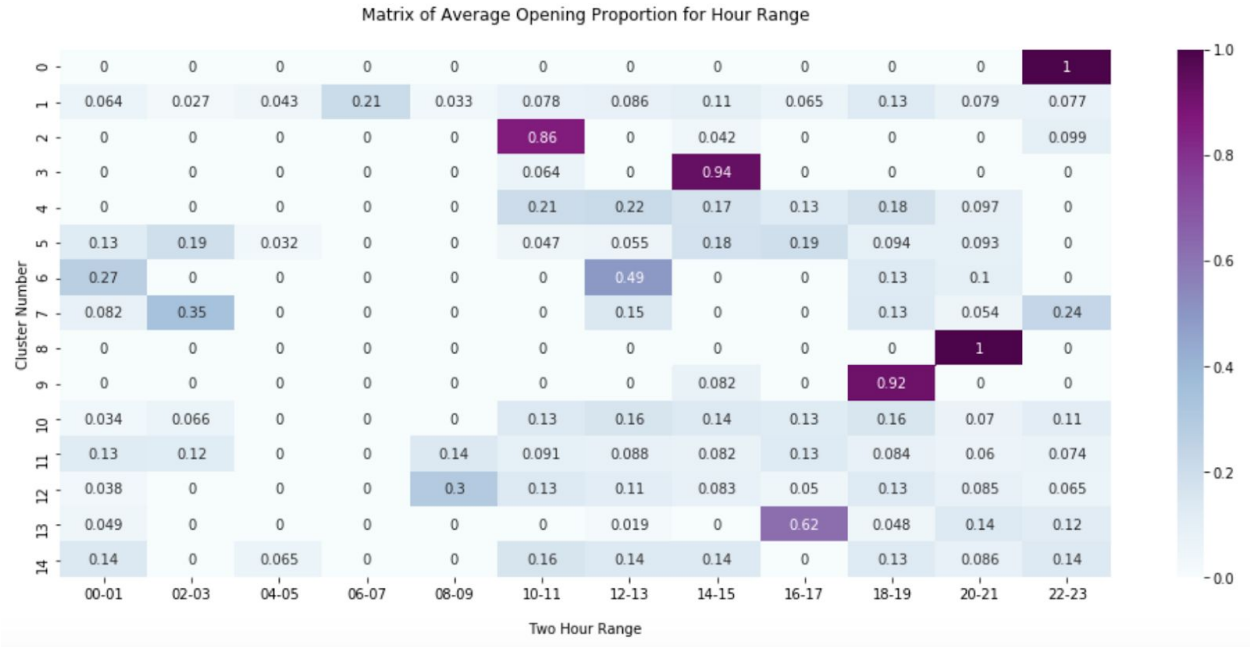
Sample of customers from Cluster 1

```
df_reg_train[df_reg_train['cluster'] == 1][0:1]
```

hour_range id	00-01	02-03	04-05	06-07	08-09	10-11	12-13	14-15	16-17	18-19	20-21	22-23	cluster
005bae737bbdee8d869d3fd39048e78891acaa3f	0.00	0.04	0.00	0.04	0.00	0.04	0.26	0.13	0.13	0.13	0.22	0.00	1

Each row represents a customer and the columns are the hour-ranges predefined. The cells are the average proportions of openings for each customer and hour-range. We can notice how much they differ from each other with concentrations of open rate in different hour ranges

Another efficient way to visualize the clusters behaviors is by the matrix below that shows the average proportion of each cluster over the hour ranges so we can check which are the most active hour range(s) for each cluster, so the highlighted proportions with darker colors represent the best hour-range for the cluster in the row.



Let's consider a random customer to illustrate how the recommendations will work in practice with the following data:

hour_range	88dea6c84f4c129d942bcc09c58fdb5e9d11088c
00-01	0.00
02-03	0.00
04-05	0.00
06-07	0.00
08-09	0.00
10-11	0.60
12-13	0.40
14-15	0.00
16-17	0.00
18-19	0.00
20-21	0.00
22-23	0.00

This data represents a customer that has opened 60% of his emails from 10 to 11 a.m and 40% from 12 to 13 (in 24 format).

So the next step will be checking which cluster the model would assign to him using **predict_proba** method so we can have his clusters probability table.

The probability table is something like image below

For each cluster defined (the numbers on the right from 0 to 14) the model predicted the customer probability and the most probable cluster for this customer is 4 and next most probable is 14 even with a very small probability.

```
[(0, 0.0),
 (1, 5.8637666770400414e-18),
 (2, 0.0),
 (3, 0.0),
 (4, 0.99999887972625112),
 (5, 3.4339519566758245e-17),
 (6, 0.0),
 (7, 0.0),
 (8, 0.0),
 (9, 0.0),
 (10, 5.0273004086965501e-09),
 (11, 4.8179994393063266e-14),
 (12, 1.367406827506057e-08),
 (13, 0.0),
 (14, 1.1015723350562132e-06)]
```

Note: *If we go back to the Matrix of Average Opening Proportion and check cluster 4, we can notice that the hour ranges 10-11 and 12-13 are the two highest average proportion so the model did a good work on assigning this cluster to this specific customer.*

Now we need to extract the next most probable cluster from **similar users** to this example customer. So let's first get people from the same cluster (similar users) and use predict_proba method again to get all the probabilities for each customer.

People within the same group that example customer:

hour_range	00-01	02-03	04-05	06-07	08-09	10-11	12-13	14-15	16-17	18-19	20-21	22-23	cluster
id													
0009b0b413429e75bf81c8989d7c6201f2d86180	0.00	0.00	0.00	0.00	0.00	0.17	0.33	0.00	0.00	0.50	0.00	0.00	4
000e522c0071795551e9ebc958ea80be87d64ec3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.33	0.00	0.00	4
0022c01e5b0850b7230f43ec05ab9b44131e5456	0.00	0.00	0.00	0.00	0.00	0.15	0.62	0.12	0.04	0.04	0.04	0.00	4
00514084cf11175e49589c542b6f358eb3895041	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00	4
005c56a6afec10a516262dd0c699d224579c377c	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.60	0.00	0.00	0.00	0.00	4

We can notice by this sample a big concentration on hour ranges 10-11 and 12-13 like the example customer data.

After predicting the most probable **next** cluster for each similar user we have a frequency table below.

	freq
14	1160
10	628
5	142
13	129
12	57

Over 2116 people from the same customer's cluster, 1160 have the cluster 14 as their next most probable cluster , followed by cluster 10 with 628 people and so on. Therefore, apart from the two most probable customer's clusters, the cluster 14 also will be selected.

WIth that, we finally have the three clusters that will be the base of recommendations and now we just have to get their hour ranges proportions.

Recap**:**
Customer Most Probable Cluster: 4
Next Customer Cluster: 14
Next Similars Cluster: 14

Which hour range would be the best recommendation?

To recommend an optimal send time for this customer and also explore new time slots, the column with the maximum average proportion of each cluster (row) will be used as the most probable hour ranges.

Customer's recommendation clusters:

hour_range	00-01	02-03	04-05	06-07	08-09	10-11	12-13	14-15	16-17	18-19	20-21	22-23
cluster												
4	0.00	0.00	0.00	0.00	0.00	0.21	0.22	0.17	0.13	0.18	0.10	0.00
14	0.14	0.00	0.07	0.00	0.00	0.16	0.14	0.14	0.00	0.13	0.09	0.14
14	0.14	0.00	0.07	0.00	0.00	0.16	0.14	0.14	0.00	0.13	0.09	0.14

The goal is to merge this rows into one single row to come up with the customer final probability table.

prob	0.00	0.00	0.00	0.00	0.00	0.21	0.22	0.17	0.13	0.18	0.10	0.00

***Note**: As the most probable cluster of this customer (which is 4) had already higher proportions than cluster 14 with maximum of 0.16 in 10-11 hour range, the final probability table is the same as the customer's most probable cluster.*

The probability table must sum to 1 in order to pass it as parameter to **numpy.random.choice** function. This function will be used to randomly select the hour range given the probability table so it can recommend and explore new time slots. Therefore this data can be normalized dividing them by the sum of proportions forcing their sum up to 1.

Finally with the final table ready, we just need to call the numpy method to randomly select the hour-range like the example below. If we run this enough times we can see the results would follow the probabilities table.

Recommender in action

```
recommended = np.random.choice(final_table_prob.index, p=probs)
print('A suggested hour range for this customer would be {0}'.format(cluster_table.columns[recommended]))
```

```
A suggested hour range for this customer would be 12-13
```

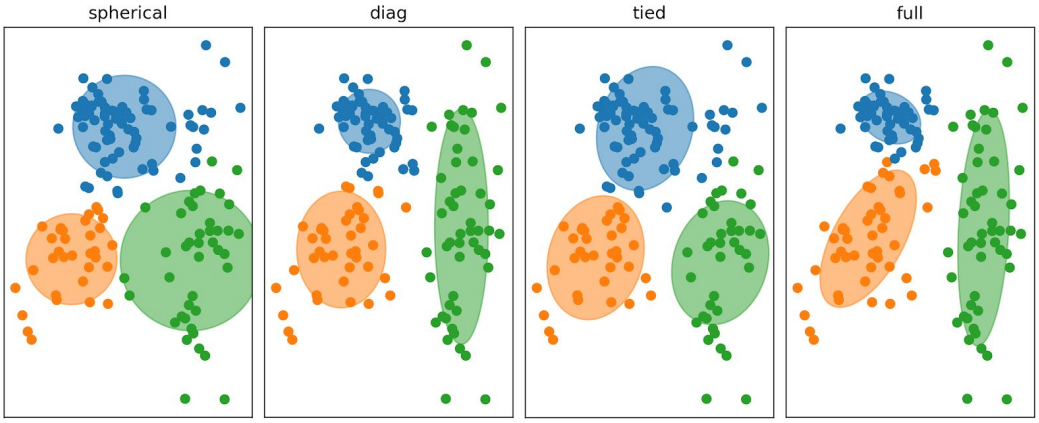
All the code for this process is implemented and properly documented in the file recommender.py which contains the class **SendTimeRecommender** with a method named **recommend_send_time** that requires 3 parameters: customer_id, target date and **learning_rate**. This last parameter is responsible to control how much we want the model to "learn"in the context of exploration of new time slots.

> ***Note:*** *This metric is inversely proportional to the probabilities of getting the most frequent user's hour range, therefore the higher this number, the more this model will learn from new time slots so we can balance this parameter according to our needs.*

Refinement

With the solution described up to this point, the initial model could suggest hour ranges given a target date and customer data but still with low F1-Score and it could be due to cluster parameters, which could be grouping people with low similarity and learning rate control.

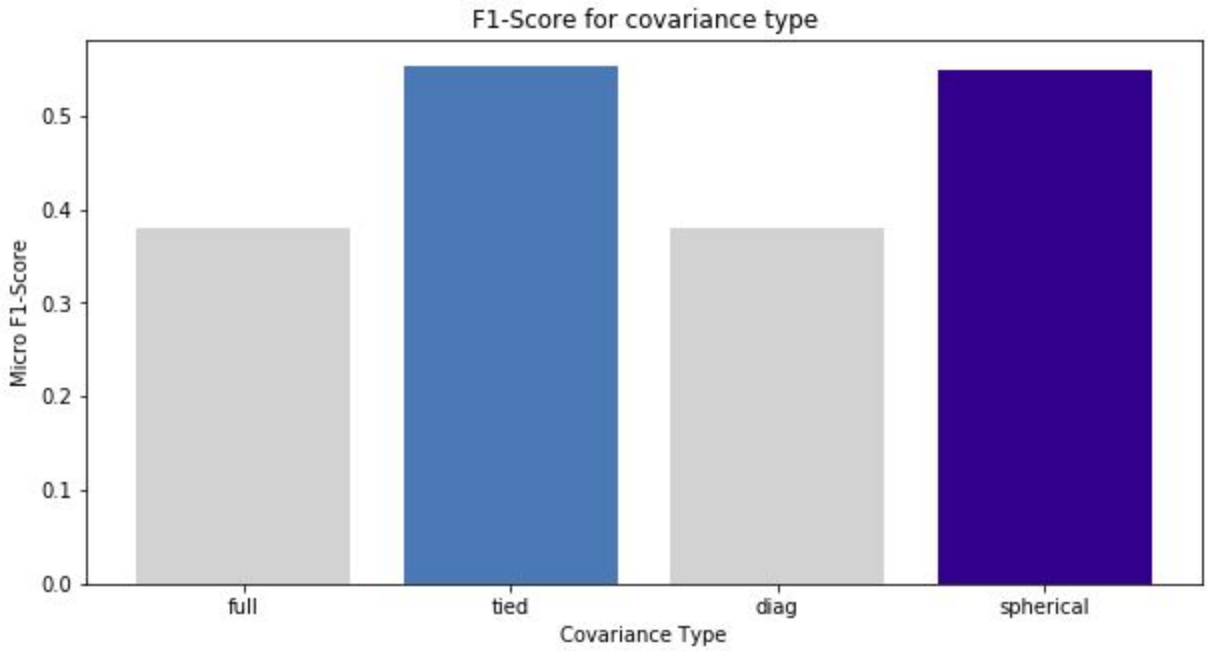
For cluster parameters refinement, the covariance type parameter of GMM cluster have been tested with 4 different values according to sklearn documentation:



covariance_type : {‘full’ (default), ‘tied’, ‘diag’, ‘spherical’}

- **‘full’:** each component has its own general covariance matrix
- **‘tied’:**all components share the same general covariance matrix
- **‘diag’:** each component has its own diagonal covariance matrix
- **‘spherical’:** each component has its own single variance

The initial result with default parameter (which is full type) had a Multiclass Micro F1-Score in the test set of X while using the tied value, it had a improvement of Y% hitting F1-Score. Check below the F1-Scores for each covariance type:



We can see that full is not the best parameter and with **spherical** covariance type we had the best score and also the lowest training time.

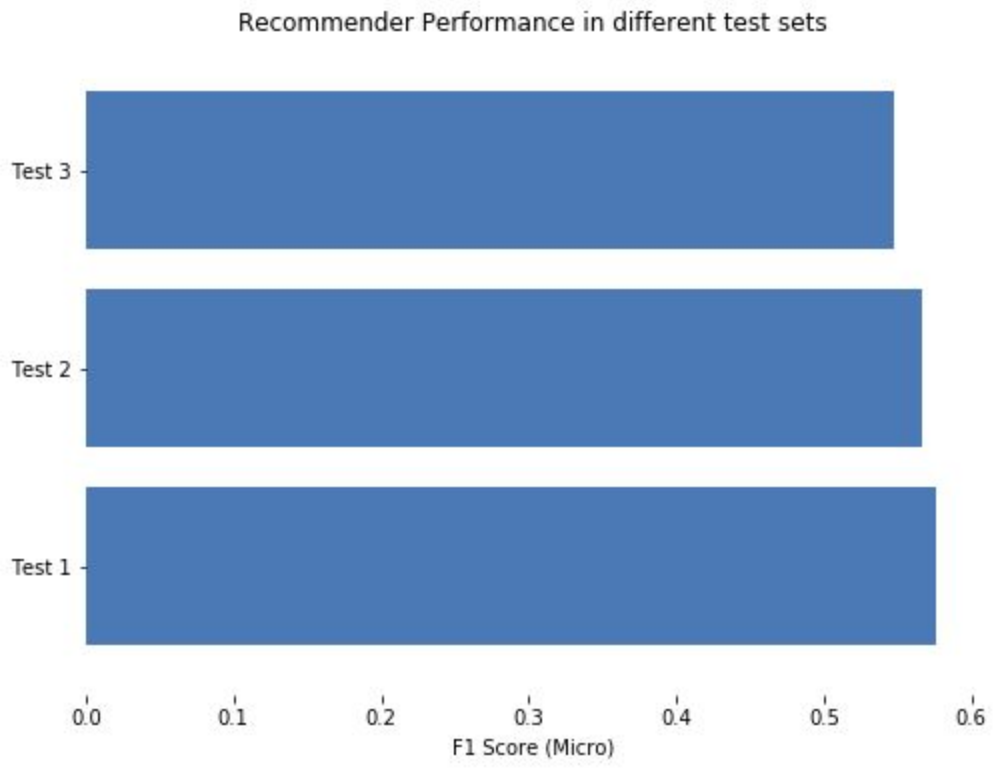
With learning rate refinement, it had a slightly change from 0.419 to 0.445 on the score using a lower value but even so, it should keep this parameter with a considerable rate of learning in order to promote explorations and exploitations. It will contribute to the recommender improvement in the future since it will be able to update its data with new information regularly and still keep the accuracy stable.

IV. Results

Model Evaluation and Validation

In order to test model's robustness and generalization capacity, some changes have been made on training and test sets so we can measure how it performs on different configurations of the dataset.

By the graph below with analysis results, we can confirm that it has reached good and similar scores to the initial tests even with different data, indicating a considerable level of generalization making it reliable at least on the dataset used for this project.



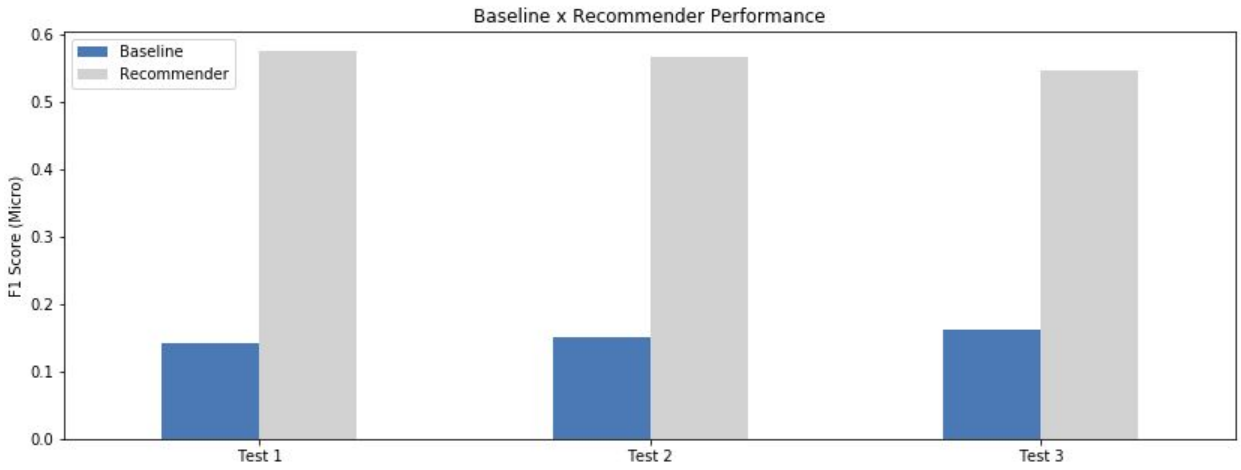
Once the test is performed on the real world with different customer's datasets from different industries like Fashion, Entertainment, Services, among others, this robustness and generalization capacity can be better evaluated.

So with the final parameters of **learning_rate** and **covariance type** previously set on the refinement section, this first version of recommender mechanism has already reached an acceptable result to keep the work going and it will be used to initially validate whether there is an evidence that the time an email is dispatched might be a factor that influences open rate.

Justification

The tests with the baseline model that uses the most frequent open time for each customer in recommendations, demonstrates it is a weak classifier.

The recommender model had twice the F1-Score value found on baseline model tests, even in different datasets configurations used to evaluate the model's sensitivity.



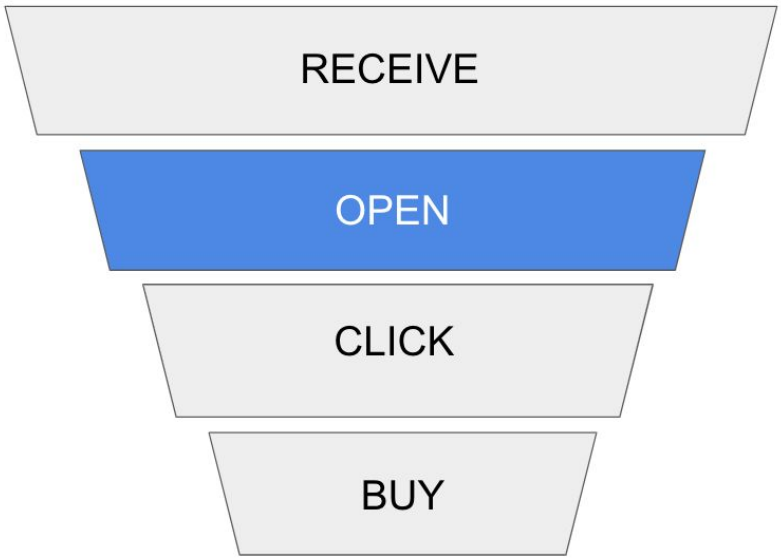
From the model performance score, we could see the capacity to recommend optimal send times in approximately 55% of the cases, so it is expected to have **some** improvement using the recommender in the whole process of digital marketing campaigns and reject the null hypothesis that send time does not influence email open rate.

With this performance we believe that it is a good start to solve this kind of problem and a smarter and more personalized way when compared to campaigns that are not based on any data driven method.

By running A/B tests it will be possible to evaluate any significant difference between optimal group (with time suggestions) and controlled group (without any recommendations mechanism) to measure how much does the time influence (or not) the openings. These results will guide the next steps for improvements if it's proved to have some relation on time and the choice of email openings.

V. Conclusion

Email marketing funnel



This is a simple Email Marketing Funnel demonstrating the general steps until a conversion (purchase).

by increasing the open rate we expected some reflections on **conversion rate**

The implicit goal of this recommendation problem is that by increasing open rate the conversion rate should be also increased and consequently the Company's revenue from Digital Marketing Campaigns. In summary we are building mechanisms to make the communication between the company's and their customers more guided to the customer's needs. So by offering the right thing to the right person **at the right time**, the customer's engagement (even when he/she does not buy anything) and the way they perceive your brand gets better and better.

So one important thing to notice in this project is that: it's not only about suggesting a probable time that a given person would open emails, **is about increasing company's influenced revenue by data driven digital actions**.

Reflection

Let's recap the steps took until here so we can interpret the outcomes properly.

We have collected data of opened emails events from Dito's CRM platform from January to November 2018. Then the data was split into train and test sets, with events until October in the training set and November in test set.

The main goal is to build a recommender that could suggest the time that each customer is more likely to open their emails in the future, meaning that with no access to November data in the training phase, the recommender should be able to predict an open time for November.

The success criteria is: after learning from past data the model should be able to suggest an hour-range that people would open an email and for this criteria it has reached this primary goal.

What factors make people open their emails?
(Or even more important to know, what prevents them to open?)

It could be due to many external factors like content of interests, the customer engagement with the brand/company, the current stage of their lives (for example male customers might not open emails about pregnancy until his wife get pregnant) and the question that drives this project is **Is the time you send an email one of the factors that could prevent them to open it when it's not a good time**?

The current process of Dito's customers have no mechanisms to suggest send times, so they are sent on random times or based on overall trends like sending always on Tuesdays at 10 A.M and we know that people have different behaviors and using generic solutions are not the best way to engage customers.

The entire process of development, adjustment and improvement was detailed in this document, so we could follow the steps since the problem framing until its results, but the real challenge is to figure out how much open rate is related to **conversion rate,** which is in fact the main goal in most of Marketing Campaigns, hence having a good mechanism to suggest optimal time does not mean that open rate or conversion rate will be positively affected.

As it has been discussed earlier, this relationship between email send time and open rate can only be tested with higher assurance using controlled tests checking the open rate of optimal group (people with best time recommended) and controlled group (people without any recommendation).

Improvement

Some improvements might be considered in this specific solution like reducing prediction time to use it with high volume of data.

Another data that might be used in future steps to suggest send time is "live" user's data, using websites events tracking to know when the user is online. Once we know a customer is online, he/she might be more likely to open emails at this time.

Furthermore there will be always a way to frame problems differently exploring others techniques, for example through **Reinforcement Learning algorithms** that could be used in an optimization perspective learning by interactions and also exploring and exploiting new time slots.

Multi Armed Bandit might be considered in the future in an attempt to increase and optimize the outcomes by testing different approaches like the one presented as a new benchmark model, maybe a classifier that could use general data and predict the probabilities for each customer opening an email for each day hour, based on demographic data and weekdays, In addition to keep with the most frequent time (the baseline model for this project), and also try the live data approach mentioned above. All these approaches and others could be used to learn by interactions collecting rewards and choose the best approach for each case.