Table of contents

1. Introduction
2. Background
3. Dataset description
4. Scope of analysis
5. Exploratory Data Analysis (EDA)
6. Derivation of user preference for email response
    Rule 1: Time difference always < 90 then he's an active person
    Rule 2: More than 70% of emails are opened at a single time window, then assign that window.
    Rule 3: Min. of opening time with > 20% of emails opened
    Rule 4: Max. percentage of emails opened from the rest of the dataset
7. Feature engineering
8. Train-test dataset split
8. Modeling
9. Productionization
10. Conclusion
11. Future work & Improvement in the model

1. Introduction

This document outlines the work for predictive modeling on email sending time window. Emails are sent with promotional/informative contents. The users have a preference of email opening time. The task is to optimize the time gap between email sending time and email opening time.

2. Background

Pair Finance has the email sending engine. Users are sent one or more emails at different times. The users have the tendency to either act within hours or they prefer to open emails at certain times e.g. after lunch or in the morning etc. If the engine has the information of user preference given users' features, the email service can be optimized very well.

3. Dataset description

The dataset has email sending times at a user level. Every user has few user level features as well as email sending and opening time. Only users having opened an email are present in the dataset. The dataset also contains the total number of emails sent to them.

4. Scope of analysis

Preliminary exploration has shown negative time values present for email delivering. Those data rows are excluded from the analysis. Also, there are data points where the email opening time window is between '0 to 6am', they should be avoided. For the modeling purpose I've included for now as there are less data points in that time window, but I'll exclude those from model in production.

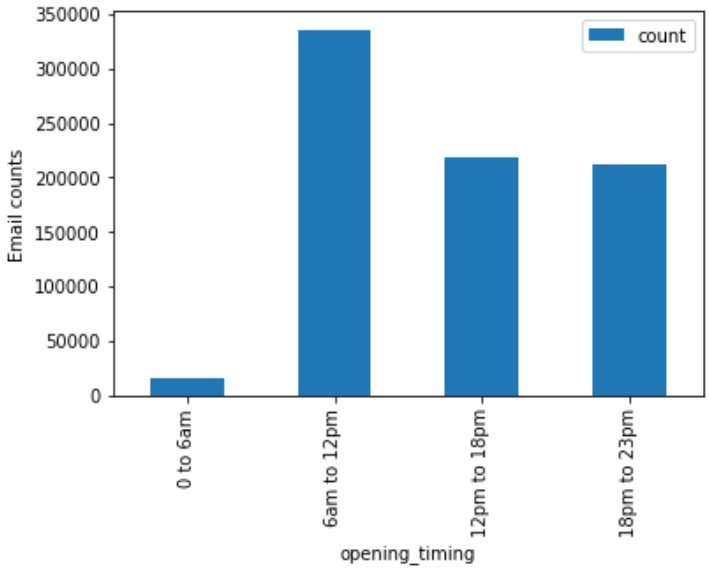
5. Exploratory Data Analysis (EDA)

Given the dataset features are very small in number, only two kinds of graphs are plot. First, the total number of emails opened in specific time window is checked. Second, number of users falling into the windows are checked. This will have repeated users falling into the same window as they've multiple entries with different email opening window.

Email level opening time window:

Table 1: Summary of email counts over time windows

opening_timing	count
0 to 6am	15389
12pm to 18pm	217668
18pm to 23pm	211425
6am to 12pm	335947

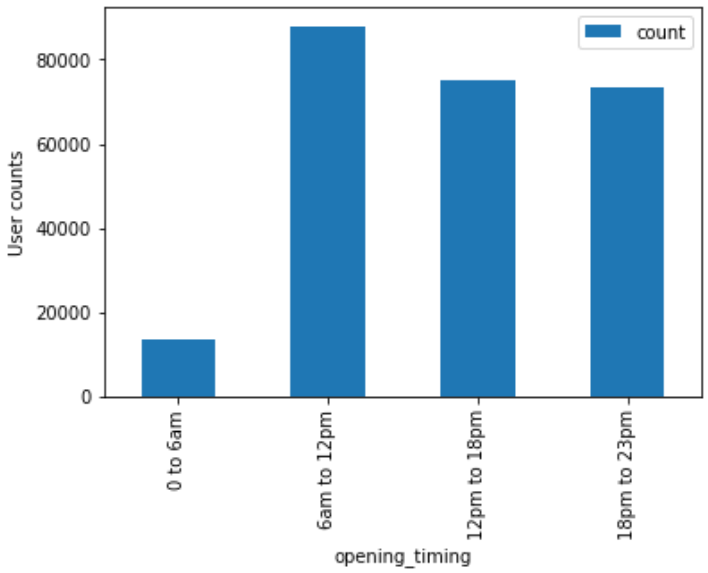


There seems to be more email opening events going on in the morning.

User level opening time window:

Table 2: Summary of user counts (repetitive) over time windows

opening_timing	count
0 to 6am	13354
6am to 12pm	88038
12pm to 18pm	88038
18pm to 23pm	73238



There seems to be more users active in the morning.

6. Derivation of user preference for email response

Given the dataset has users with different email opening times, the predictive model is directed towards optimizing the opening time. For this purpose, every user is assigned a single time window by means of a top down funnel approach. There are 4 rules defined for this purpose. The users are assigned sequentially through this method. There are a total of 100000 users in the raw dataset with 99851 users included for the analysis.

Rule 1: Time difference always < 90 then he's an active person

This rule contains a total of 18230 users. Those users are further assigned a preferred time window based on the number of emails opened in each window.

Rule 2: More than 70% of emails are opened at a single time window, then assign that window.

This rule has 22566 users.

Rule 3: Min. of opening time with > 20% of emails opened

From the remaining users in the dataset, the users are filtered based on minimum opening time given there are more than 20% transactions available for the corresponding transactions.

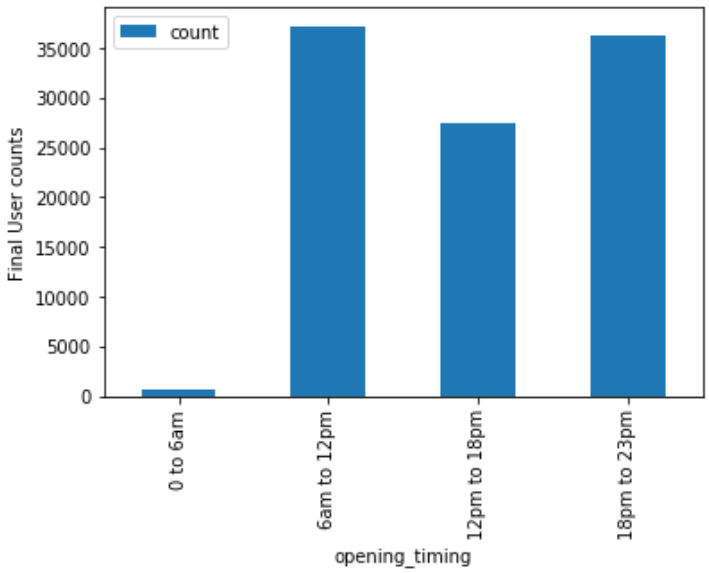
Rule 4: Max. percentage of emails opened from the rest of the dataset

The final time window assignment is based on the maximum number of emails opened.

Final user level opening time window:

Table 3: Summary of user counts (non-repetitive) over time windows

opening_timing	count
0 to 6am	680
6am to 12pm	37254
12pm to 18pm	27423
18pm to 23pm	36194



Users seem to be preferring morning and evening session for opening of emails.

7. Feature engineering

The feature is divided based on the data types. The user features has both numerical and categorical entries.

Numerical features [‘X1’,’X2’]: Though the provided data did not have any Null values, SimpleImputer is used to replace the values with median values of the corresponding column.

Low cardinal features [‘X3’]: Categorical variables having less number (4) of distinct categories in it are converted to dummy variables using one-hot encoder.

8. Train-test dataset split

The dataset is divided randomly into train and test dataset. The train dataset is further divided into train and validation dataset.

Table 4: Train-test dataset split

Segment	Total volume	% Population
Train	79880	80
Validation	9985	10
Test	9986	10

8. Modeling

The modeling is carried out using an XGBoost model. Linear regressor is used for the objective function. Though there wasn't any need to use the relatively complex model like XGBoost, but the easy implementation of the algorithm makes it the choice of use.

The model now predicts the email opening time in minutes given the features.

Table 5: Final modeling dataset

X1	X2	X3	TO_minute
-1.25	-0.88	2	1263
-0.14	-3.21	0	1031
1.23	0.08	3	562
-0.38	-1.3	3	758
-1.5	0.22	2	703

Input variables are: ['X1','X2','X3']
Target variable: ['TO_minute']

Performance on the test dataset:

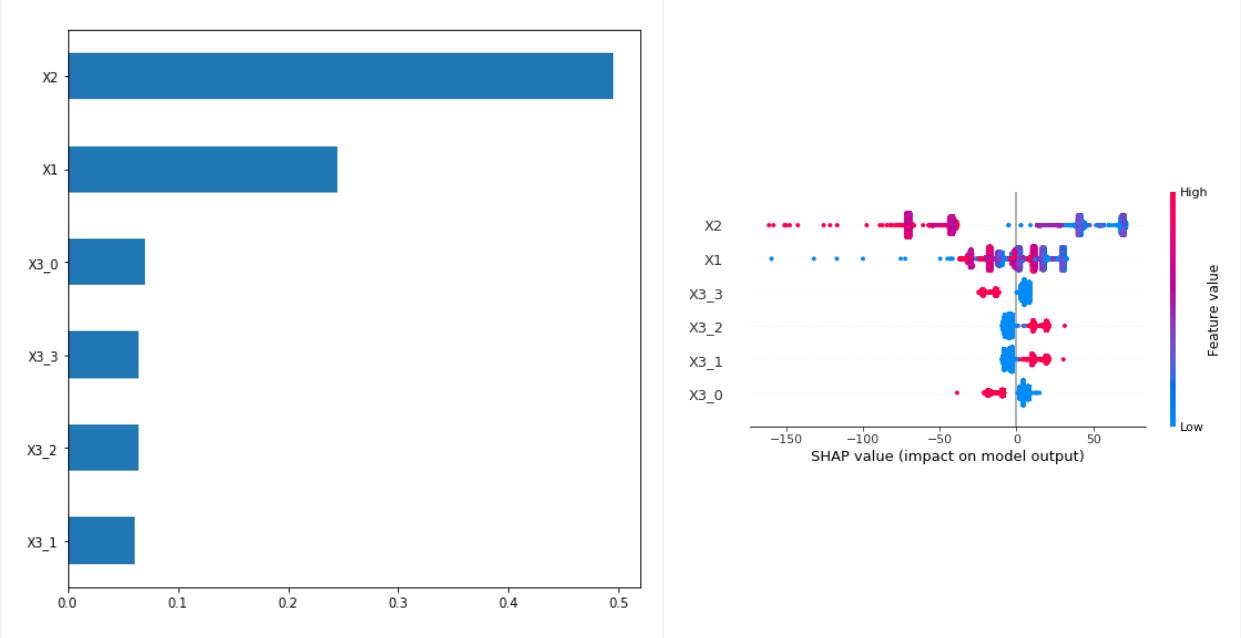
Table 6: Performance of the model in minutes

Train RMSE	Validation RMSE	Test RMSE
296.57	294.60	296.37

Most important variables:

Table 7: Most important variable plot

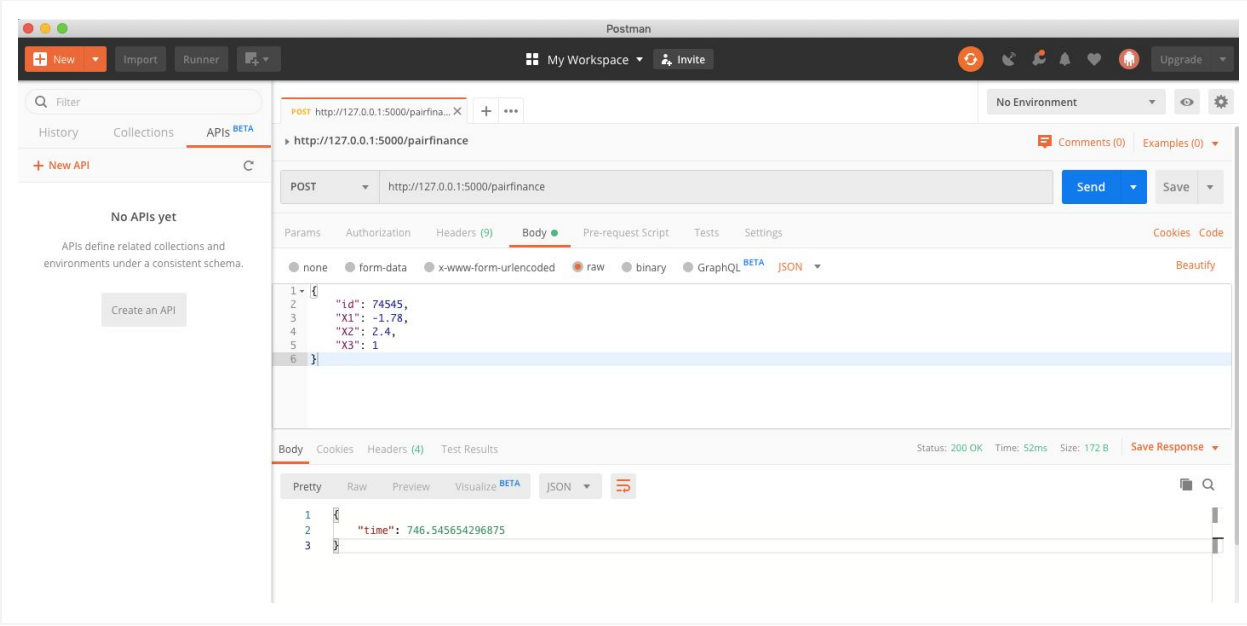
Feature importance (XGBoost)	Shap values



9. Productionization

The package is created inclusive of the model. The model is served locally using flask module in python.

Table 8: Postman query example



10. Conclusion

Given the data, the model predicts a wide range for email sending time window. However, there are many improvements on top of this model as mentioned below.

11. Future work & Improvement in the model

1. The modeling is carried on based on the defined logic for user preference on email reply. This can be fine tuned further.
2. There can be 4 individual models built for specific group of people.
3. More data and more features will surely increase the accuracy.
4. Hyperparameter tuning and different ML models can become handy while exploring alternative better solution.