Objective 1: What percentage of users opened the email and what percentage of users clicked on the link inside the email?

To figure this out, lets move ahead step by step:
First we try and figure out the no of entries in our dataset
By analyising our dataset we found that:-

```
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   email_id             100000 non-null  int64 
 1   email_text           100000 non-null  object
 2   email_version        100000 non-null  object
 3   hour                 100000 non-null  int64 
 4   weekday              100000 non-null  object
 5   user_country         100000 non-null  object
 6   user_past_purchases  100000 non-null  int64 
dtypes: int64(3), object(4)
```

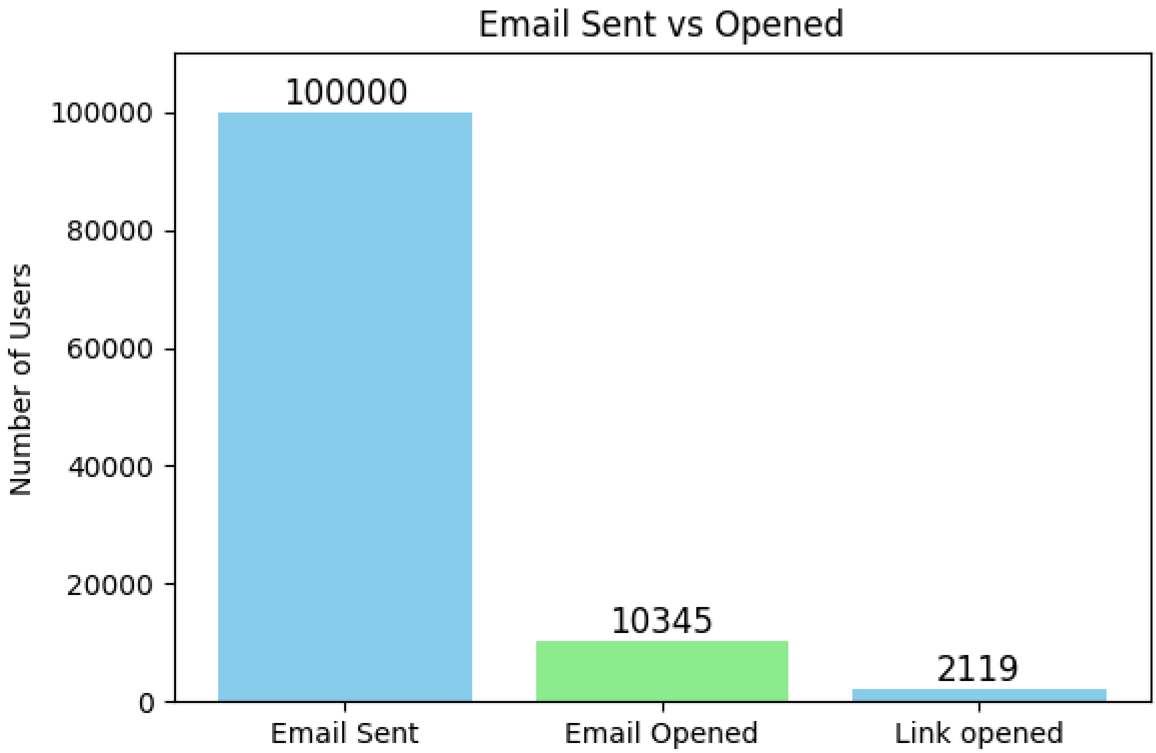
There were 100000 entries in total with no duplicate email ids, the email_text, email_version, weekday and user_country were string data types (referred to as object of characters here) and the rest were integer data types.
Next up, i merged all the tables into one for better and simpler analysis

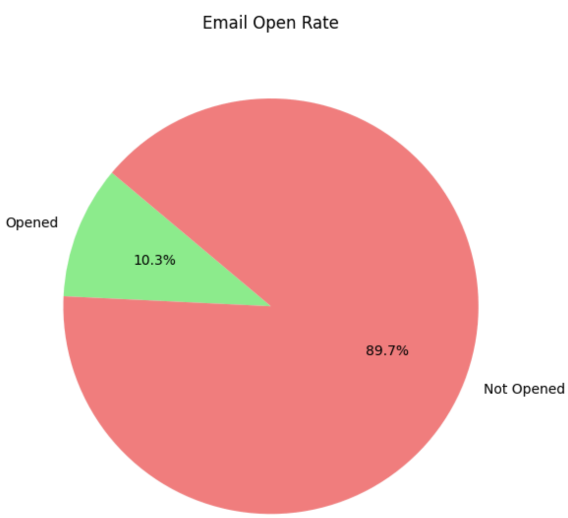
```
[ ]  df_emailtable['Email opened'] = df_emailtable['email_id'].isin(df_emailopened['email_id']).map({True: 'Yes', False: 'No'})

[ ]  df_emailtable['Link opened'] = df_emailtable['email_id'].isin(df_linkopened['email_id']).map({True: 'Yes', False: 'No'})
```

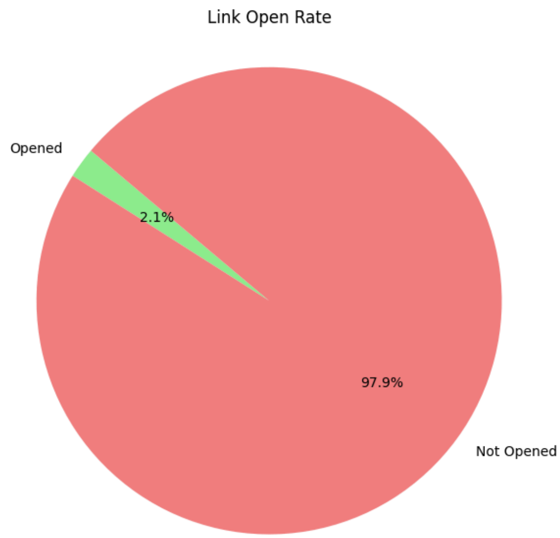
Resulting table looked like this

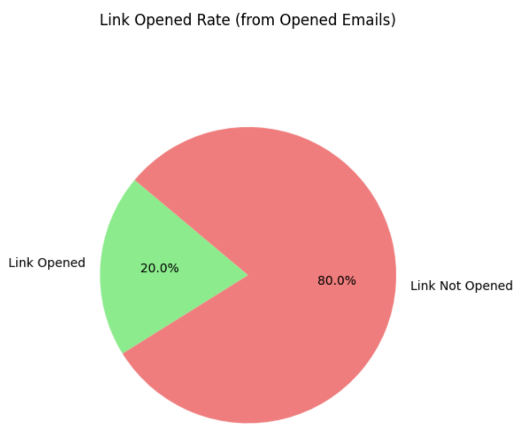
	email_id	email_text	email_version	hour	weekday	user_country	user_past_purchases	Email opened	Link opened
0	85120	short_email	personalized	2	Sunday	US	5	No	No
1	966622	long_email	personalized	12	Sunday	UK	2	Yes	Yes
2	777221	long_email	personalized	11	Wednesday	US	2	No	No
3	493711	short_email	generic	6	Monday	UK	1	No	No
4	106887	long_email	generic	14	Monday	US	6	No	No

This made very easy to analyse the number of people in total, the number of people who opened the email and the number of people who clicked on the link inside the email









Thus cumulatively getting a Conversion rate of: 2.12% and an Open Rate of 10.35%

Q2)

As part of our effort to optimize email strategies and move away from random dispatching, I developed and tested multiple machine learning models aimed at predicting whether users would click on links embedded in emails. The key features used included: email_text, email_version, hour, weekday, user_country, user_past_purchases, and Email opened. I also used feature engineering techniques and made new features like working hours and classed user past purchases to identify types of customers

Despite extensive experimentation—including model selection (Random Forests, Gradient Boosting, Logistic Regression), hyperparameter tuning, feature engineering, and class imbalance handling with SMOTE—the results consistently showed high accuracy but very low precision and recall for the minority class (i.e., users who actually clicked the link).

Key Observations:

- The dataset is highly imbalanced, with the “Link opened” class (positive class) being very sparse and likely influenced by random, non-observable human behavior.

- Most models ended up overfitting to the majority class (i.e., predicting users would not click), due to the limited signal in the minority class.

- I have included confusion matrix screenshots from various models and the code is documented in the attached Google Colab notebook.

Strategic Insight:

While the model struggles to accurately predict who will click a link, it is significantly better at identifying who will not. This gives us a powerful "exclusion-based" strategy:

Instead of targeting everyone, we can use the model to filter out users who are unlikely to engage and focus campaigns on the remaining audience.

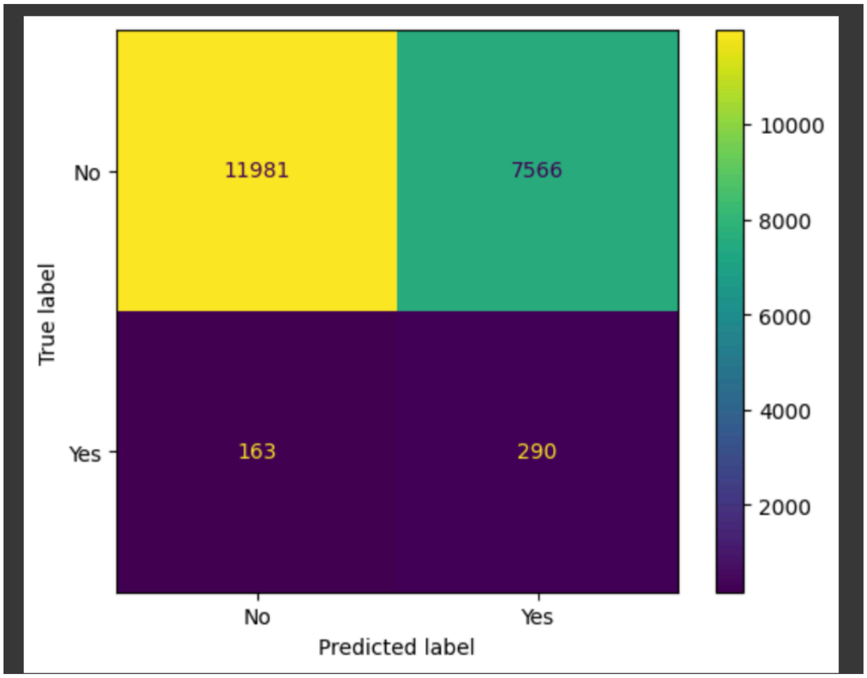
This alternative approach can help:

- Reduce email fatigue and unsubscribe rates

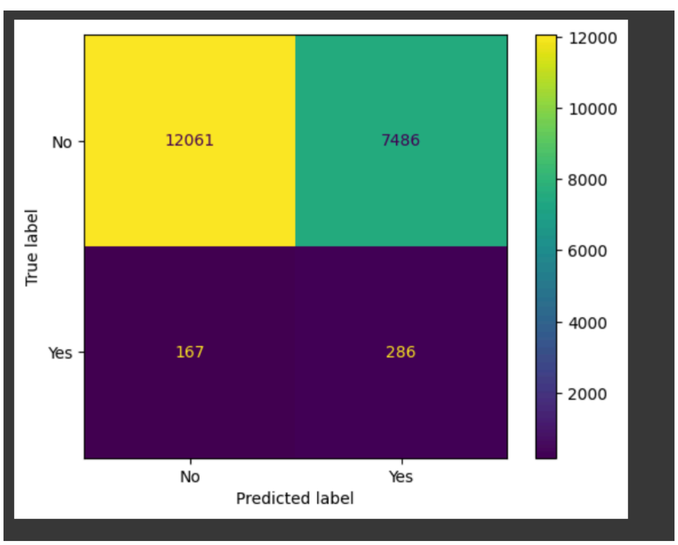
- Increase targeting efficiency

- Improve ROI by focusing on higher-probability converters

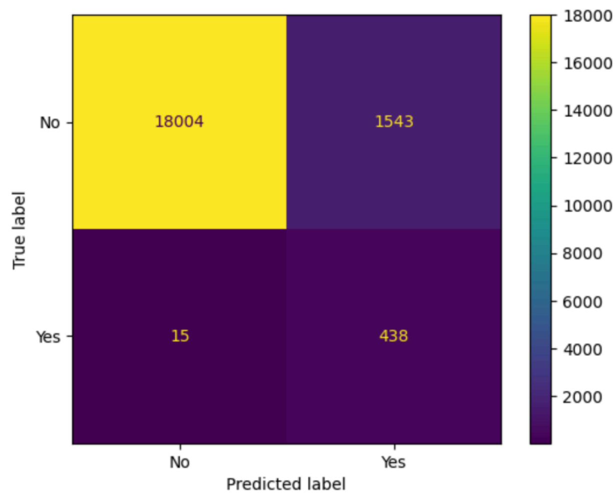
Decision trees



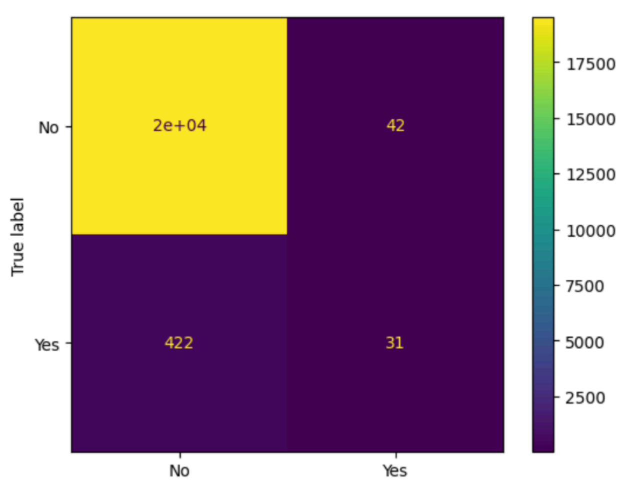
Random Forest



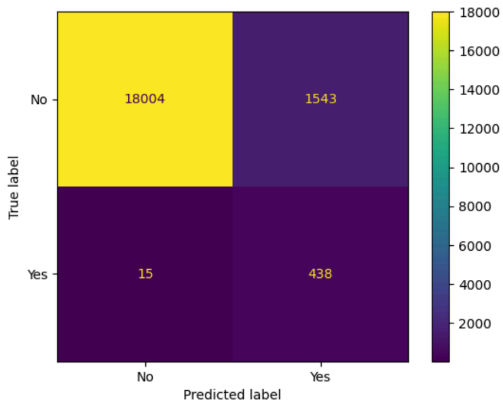
Gradient Boosting



Knn



Svm

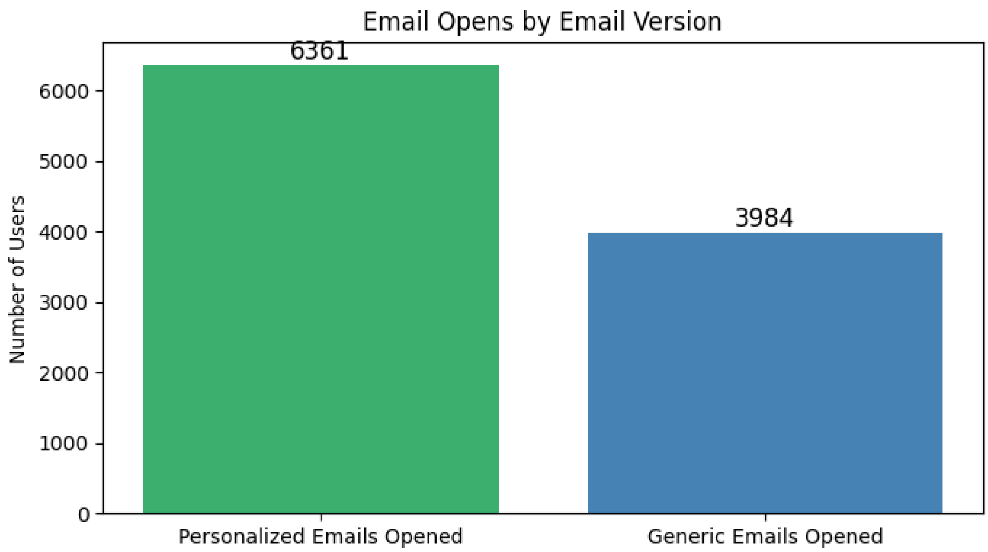


Most of the models ended up giving very high accuracy as they mostly end up predicint no which is the majority class

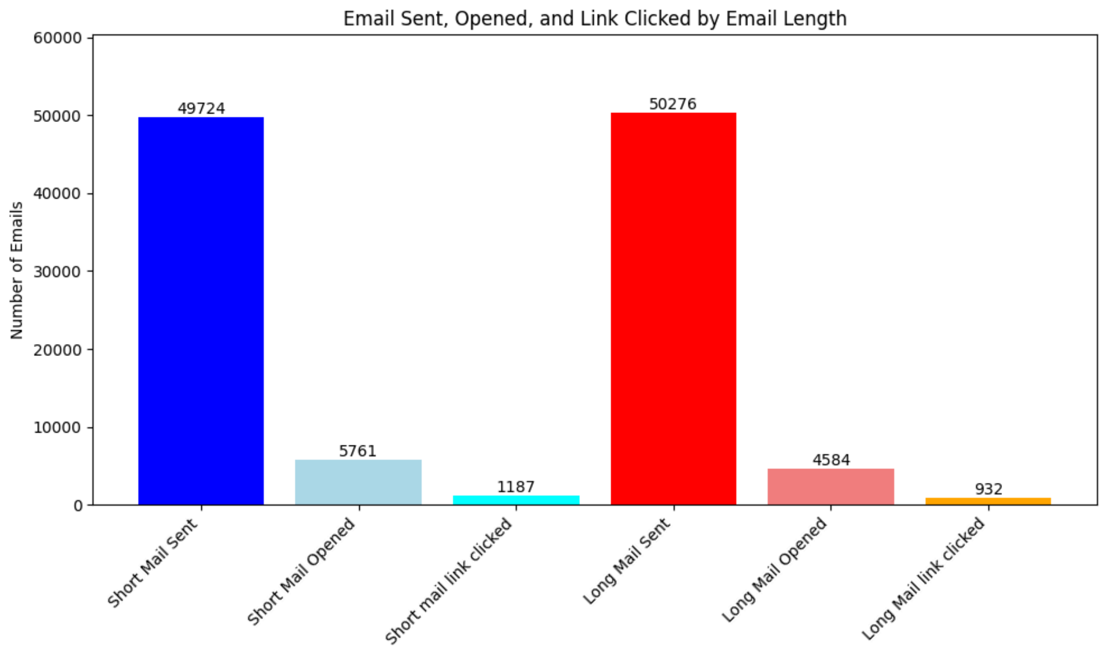
Objective 3)

Inspite of this i have framed the most optimal instances when it would be best to send mails to customers based on the given data

Sending personalized emails gives us a 12.78% chane of email being opened whereas generic gives a 7.9% chance

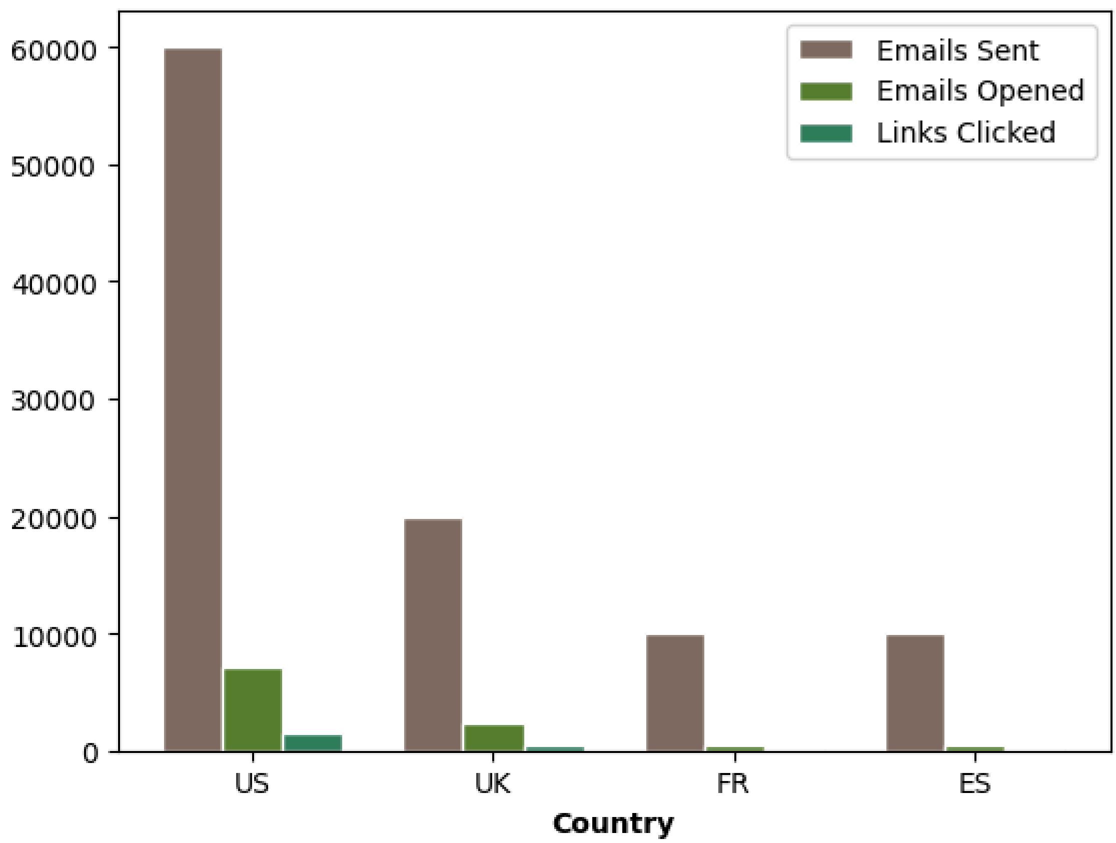


A little more short mails were opened inspite of being lesser in number as users are lesser likely to be bored

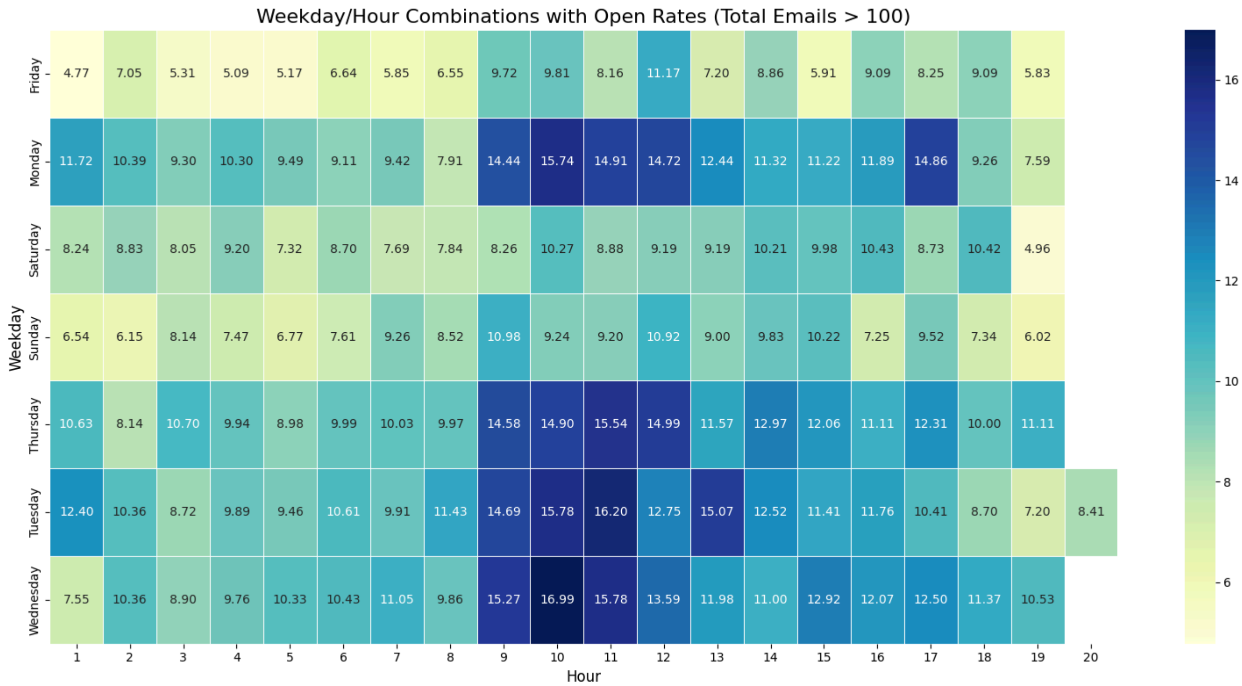


	Country	Emails Sent	Emails Opened	Links Clicked	Open Rate	Conversion Rate
0	US	60099	7153	1464	11.902028	2.435981
1	UK	19939	2396	492	12.016651	2.467526
2	FR	9995	406	80	4.062031	0.800400
3	ES	9967	390	83	3.912913	0.832748

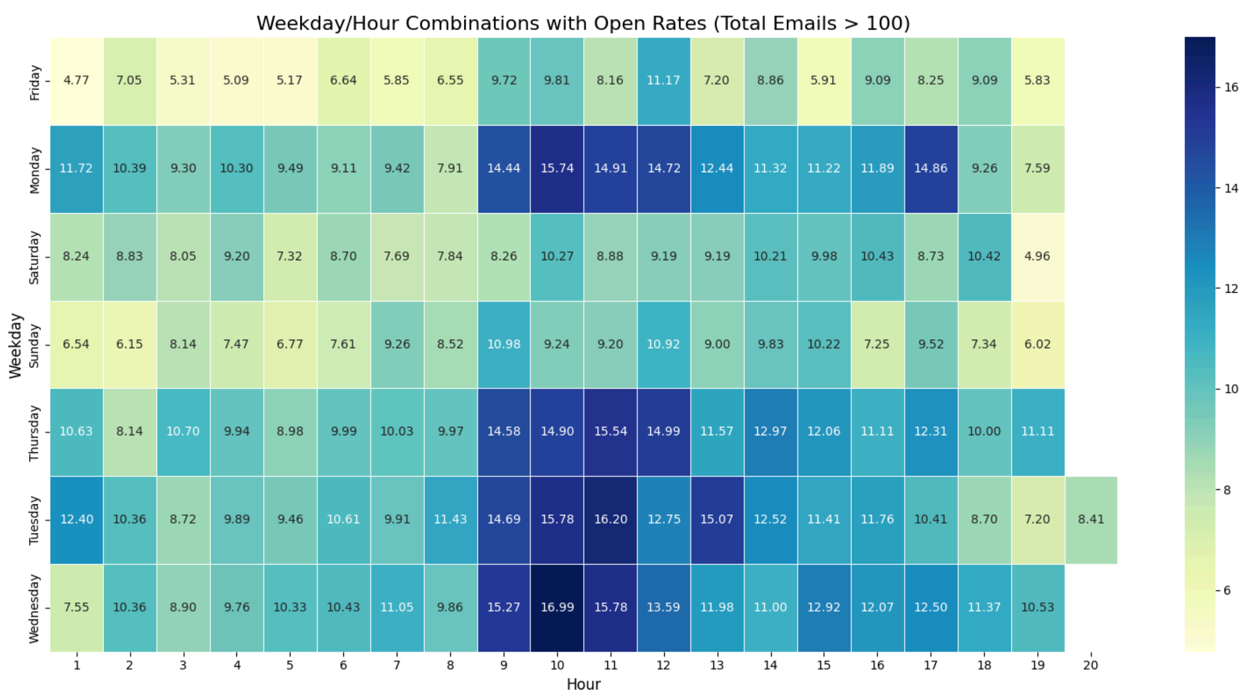
UK, had the highest open and conversion rate meaning the target audience should be Uk followed closely by US



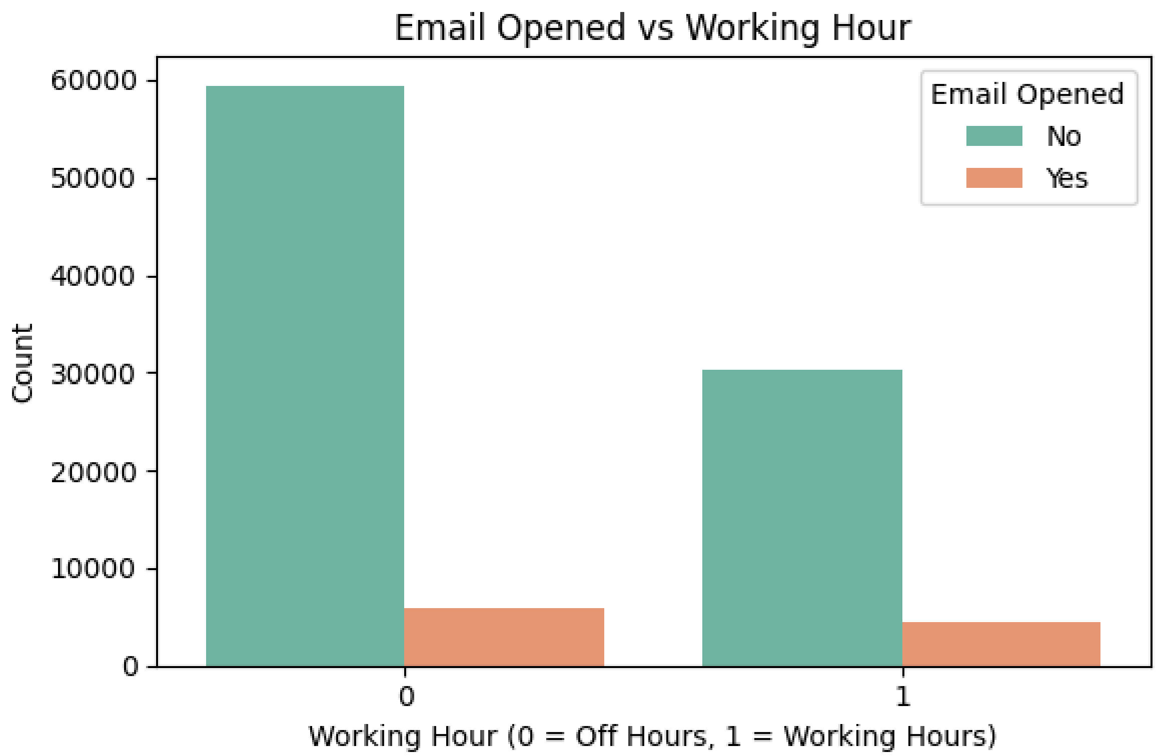
This is when users are most likely to open the mails, i noticed by far the most randomness in this instance unable to understand the reason for this trend, maybe the customers are not necessarily working 9-5s



Same observation for link open rates



My previous claim seems to be backed up by this feature engineered graph.



I even used deep learning with as many as 20 epochs but still didnt get any good meaningful patterns to accurately predict links being opened

But i was able to predict the optimal combination for maximum link open probability

```
The optimal combination for maximum link open probability is:
('long_email', 'generic', np.int64(2), 'Monday', 'UK', np.int64(10))
```

These instances had the most link opened cases.

Observation 4) According to me, the user demographic is not necessarily working as the working hour correlation is quite less, the data seems quite random at predicting when links are actually opened, this resonates well with the theme that Ai enhanced marketing is not an easy task and i am very excited to learn and improve my skills