Exploratory Data Analysis

Colab link

https://colab.research.google.com/drive/1sHLEe\_GamjQZWAGm3rDOayo3OEkd0gSr?usp=sharing

**Objective:**

To Perform exploratory data analysis on email data set.

**Data description:**

This is a email data set. Exporting all mails as data set, importing them inside a pandas data frame, visualize them and get different insights from the data.

**Program Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

**Importing Email**

import mailbox

mboxfile = "/content/All mail Including Spam and Trash.mbox"

mbox = mailbox.mbox(mboxfile)

mbox

**Accessing**

import csv

with open('mailbox.csv', 'w') as outputfile:

 writer = csv.writer(outputfile)

 writer.writerow(['subject','from','date','to','label','thread'])

 for message in mbox:

   writer.writerow([

       message['subject'],

       message['from'],

       message['date'],

       message['to'],

       message['X-Gmail-Labels'],

       message['X-GM-THRID']

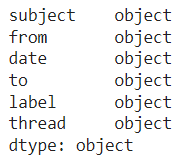
   ]

 )

dfs = pd.read\_csv('mailbox.csv', names=['subject', 'from', 'date', 'to',

'label', 'thread'])

dfs.dtypes



-----------------------------------------------------------------------

dfs['date'] = dfs['date'].apply(lambda x: pd.to\_datetime(x,

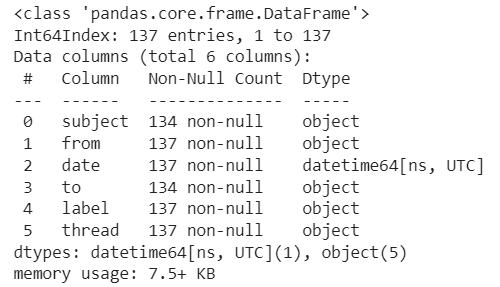
errors='coerce', utc=True))

dfs = dfs[dfs['date'].notna()]

dfs.to\_csv('gmail.csv')

-----------------------------------------------------------------------

dfs.info()



-----------------------------------------------------------------------

dfs.head(10)



import re

def extract\_email\_ID(string):

 email = re.findall(r'<(.+?)>', string)

 if not email:

   email = list(filter(lambda y: '@' in y, string.split()))

 return email[0] if email else np.nan

dfs['from'] = dfs['from'].apply(lambda x: extract\_email\_ID(x))

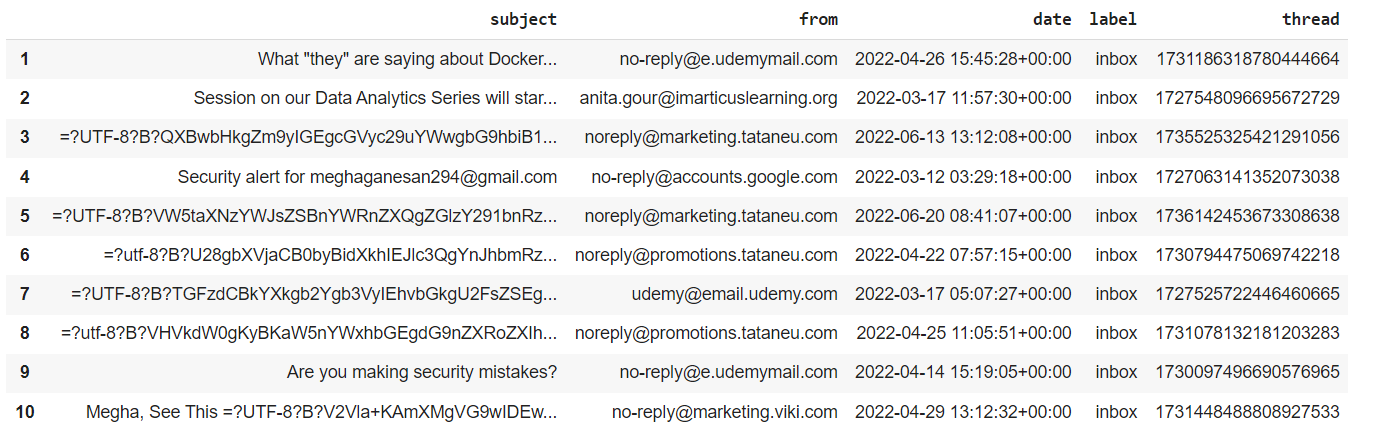
myemail = 'gmegha294@@gmail.com'

dfs['label'] = dfs['from'].apply(lambda x: 'sent' if x==myemail

else 'inbox')

dfs.drop(columns='to', inplace=True)

dfs.head(10)



-----------------------------------------------------------------------

**Refactoring timezones**

import datetime

import pytz

def refactor\_timezone(x):

 est = pytz.timezone('US/Eastern')

 return x.astimezone(est)

#convert the day of the week variable into the name of the day, as in, Saturday, Sunday, and so on

dfs['dayofweek'] = dfs['date'].apply(lambda x: x.day\_name())

dfs['dayofweek'] = pd.Categorical(dfs['dayofweek'], categories=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday','Saturday', 'Sunday'], ordered=True)

#SAME PROCESS FOR THE TIME OF THE DAY

dfs['timeofday'] = dfs['date'].apply(lambda x: x.hour + x.minute/60

+ x.second/3600)

dfs['hour'] = dfs['date'].apply(lambda x: x.hour)

dfs['year\_int'] = dfs['date'].apply(lambda x: x.year)

x.dayofyear/365.25)

dfs.index = dfs['date']

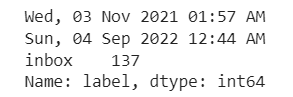
-----------------------------------------------------------------------

**Number of Emails**

print(dfs.index.min().strftime('%a, %d %b %Y %I:%M %p'))

print(dfs.index.max().strftime('%a, %d %b %Y %I:%M %p'))

print(dfs['label'].value\_counts())

****

sent = dfs[dfs['label']=='sent']

received = dfs[dfs['label']=='inbox']

import matplotlib.pyplot as plt

from matplotlib.ticker import MaxNLocator

from scipy import ndimage

import matplotlib.gridspec as gridspec

import matplotlib.patches as mpatches

def plot\_todo\_vs\_year(df, ax, color='C0', s=0.5, title=''):

 ind = np.zeros(len(df), dtype='bool')

 est = pytz.timezone('US/Eastern')

 df[~ind].plot.scatter('year', 'timeofday', s=s, alpha=0.6, ax=ax,

color=color)

 ax.set\_ylim(0, 24)

 ax.yaxis.set\_major\_locator(MaxNLocator(8))

 ax.set\_yticklabels([datetime.datetime.strptime(str(int(np.mod(ts,

24))), "%H").strftime("%I %p") for ts in ax.get\_yticks()]);

 ax.set\_xlabel('')

 ax.set\_ylabel('')

 ax.set\_title(title)

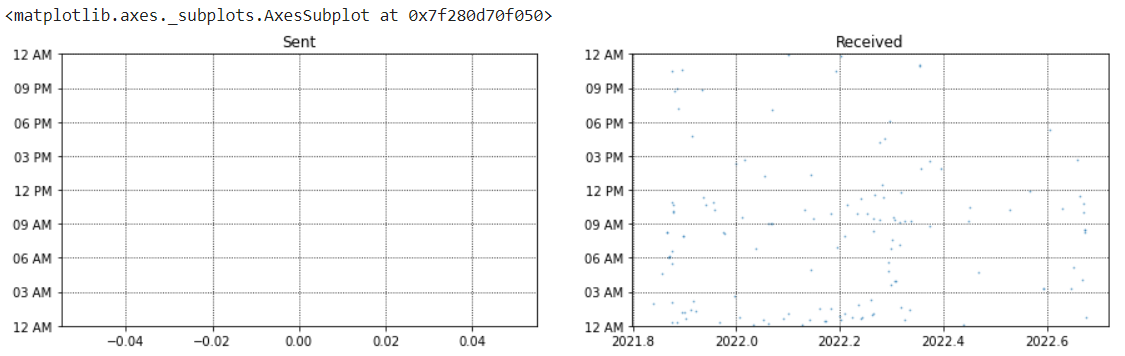
 ax.grid(ls=':', color='k')

 return ax

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15, 4))

plot\_todo\_vs\_year(sent, ax[0], title='Sent')

plot\_todo\_vs\_year(received, ax[1], title='Received')

****

**OBSERVATION:**Ihave more received mails than sent mail.

**creates a function that plots the average number of emails per day**

def plot\_number\_perdhour\_per\_year(df, ax, label=None, dt=1,smooth=False,weight\_fun=None, \*\*plot\_kwargs):

  tod = df[df['timeofday'].notna()]['timeofday'].values

  year = df[df['year'].notna()]['year'].values

  Ty = year.max() - year.min()

  T = tod.max() - tod.min()

  bins = int(T / dt)

  if weight\_fun is None:

    weights = 1 / (np.ones\_like(tod) \* Ty \* 365.25 / dt)

  else:

    weights = weight\_fun(df)

  if smooth:

    hst, xedges = np.histogram(tod, bins=bins,weights=weights);

    x = np.delete(xedges, -1) + 0.5\*(xedges[1] - xedges[0])

    hst = ndimage.gaussian\_filter(hst, sigma=0.75)

    f = interp1d(x, hst, kind='cubic')

    x = np.linspace(x.min(), x.max(), 10000)

    hst = f(x)

    ax.plot(x, hst, label=label, \*\*plot\_kwargs)

  else:

    ax.hist(tod, bins=bins, weights=weights, label=label,\*\*plot\_kwargs);

  ax.grid(ls=':', color='k')

  orientation = plot\_kwargs.get('orientation')

  if orientation is None or orientation == 'vertical':

    ax.set\_xlim(0, 24)

    ax.xaxis.set\_major\_locator(MaxNLocator(8))

    ax.set\_xticklabels([datetime.datetime.strptime(str(int(np.mod(ts,24))), "%H").strftime("%I %p") for ts in ax.get\_xticks()]);

  elif orientation == 'horizontal':

    ax.set\_ylim(0, 24)

    ax.yaxis.set\_major\_locator(MaxNLocator(8))

  ax.set\_yticklabels([datetime.datetime.strptime(str(int(np.mod(ts, 24))), "%H").strftime("%I %p") for ts in ax.get\_yticks()])

def plot\_number\_perday\_per\_year(df, ax, label=None, dt=0.3,

\*\*plot\_kwargs):

 year = df[df['year'].notna()]['year'].values

 T = year.max() - year.min()

 bins = int(T / dt)

 weights = 1 / (np.ones\_like(year) \* dt \* 365.25)

 ax.hist(year, bins=bins, weights=weights, label=label,

\*\*plot\_kwargs);

 ax.grid(ls=':', color='k')

class TriplePlot:

 def \_\_init\_\_(self):

   gs = gridspec.GridSpec(6, 6)

   self.ax1 = plt.subplot(gs[2:6, :4])

   self.ax2 = plt.subplot(gs[2:6, 4:6], sharey=self.ax1)

   plt.setp(self.ax2.get\_yticklabels(), visible=False);

   self.ax3 = plt.subplot(gs[:2, :4])

   plt.setp(self.ax3.get\_xticklabels(), visible=False);

 def plot(self, df, color='darkblue', alpha=0.8, markersize=0.5,yr\_bin=0.1, hr\_bin=0.5):

   plot\_todo\_vs\_year(df, self.ax1, color=color, s=markersize)

   plot\_number\_perdhour\_per\_year(df, self.ax2, dt=hr\_bin,

color=color, alpha=alpha, orientation='horizontal')

   self.ax2.set\_xlabel('Average emails per hour')

   plot\_number\_perday\_per\_year(df, self.ax3, dt=yr\_bin,

color=color, alpha=alpha)

   self.ax3.set\_ylabel('Average emails per day')

**Instantiate the class to plot the graph:**

import matplotlib.gridspec as gridspec

import matplotlib.patches as mpatches

import datetime

plt.figure(figsize=(12,12));

tpl = TriplePlot()

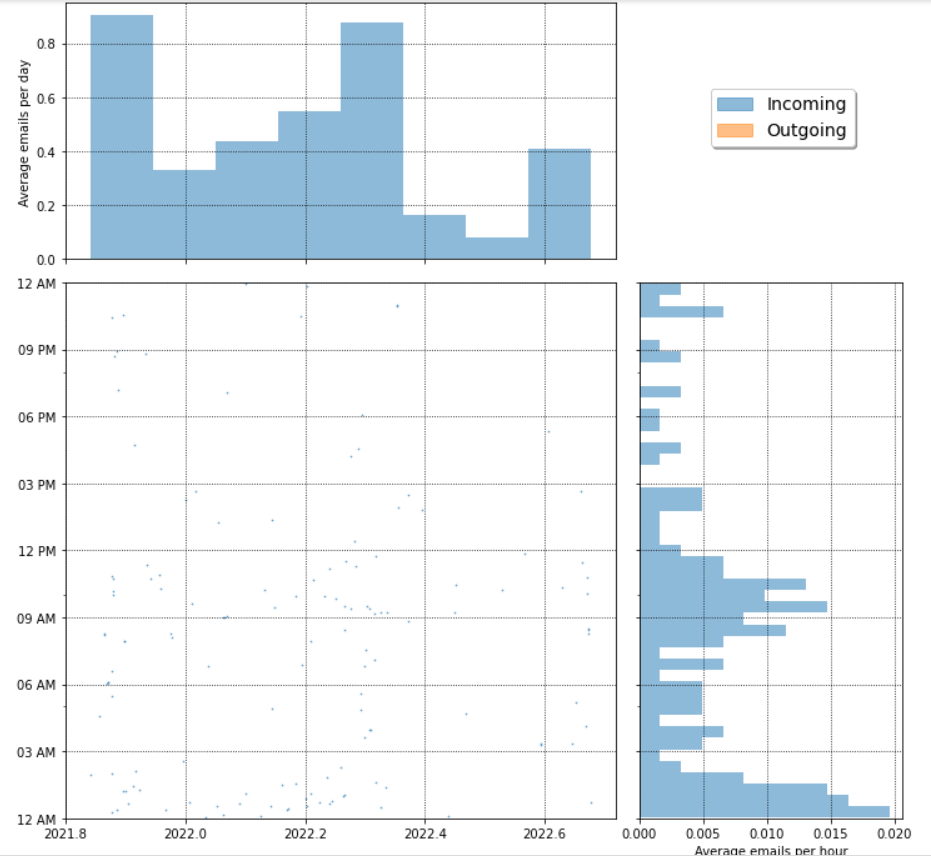
tpl.plot(received, color='C0', alpha=0.5)

#tpl.plot(sent, color='C1', alpha=0.5)

p1 = mpatches.Patch(color='C0', label='Incoming', alpha=0.5)

p2 = mpatches.Patch(color='C1', label='Outgoing', alpha=0.5)

plt.legend(handles=[p1, p2], bbox\_to\_anchor=[1.45, 0.7],fontsize=14, shadow=True);

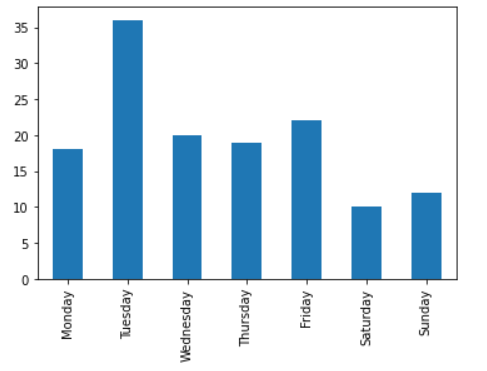


**OBSERVATION**:I having more incoming mail than outgoing mails

Number of emails per day

counts = dfs.dayofweek.value\_counts(sort=False)

counts.plot(kind='bar')



**OBSERVATION:** Tuesday has more no of mails and Saturday have least mails.

-----------------------------------------------------------------------

sdw = sent.groupby('dayofweek').size() / len(sent)

rdw = received.groupby('dayofweek').size() / len(received)

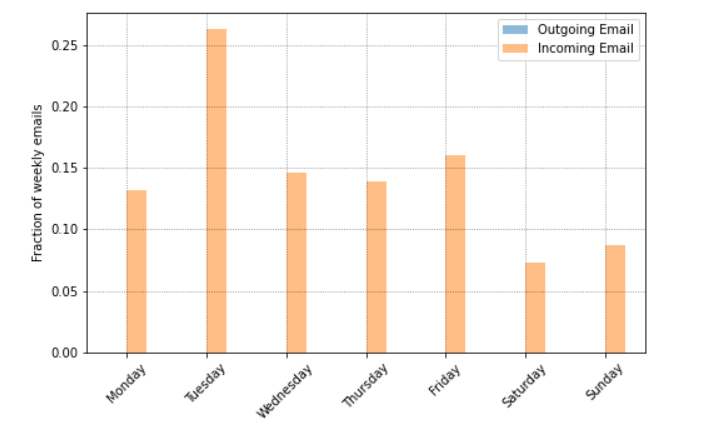
df\_tmp = pd.DataFrame(data={'Outgoing Email': sdw, 'Incoming Email':rdw})

df\_tmp.plot(kind='bar', rot=45, figsize=(8,5), alpha=0.5)

plt.xlabel('');

plt.ylabel('Fraction of weekly emails');

plt.grid(ls=':', color='k', alpha=0.5)



**OBERVATION:** There is no outgoing mails

import scipy.ndimage

from scipy.interpolate import interp1d

import datetime

plt.figure(figsize=(8,5))

ax = plt.subplot(111)

for ct, dow in enumerate(dfs.dayofweek.cat.categories):

  df\_r = received[received['dayofweek']==dow]

  weights = np.ones(len(df\_r)) / len(received)

  wfun = lambda x: weights

  plot\_number\_perdhour\_per\_year(df\_r, ax, dt=1, smooth=True, color=f'C{ct}', alpha=0.8, lw=3, label=dow, weight\_fun=wfun)

  df\_s = sent[sent['dayofweek']==dow]

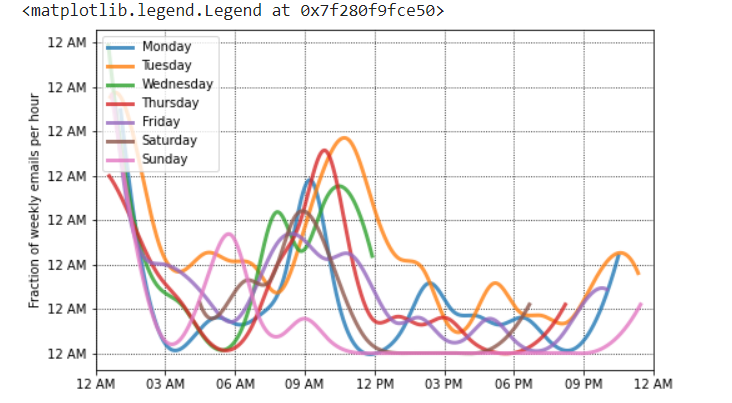
  weights = np.ones(len(df\_s)) / len(sent)

  wfun = lambda x: weights

  #plot\_number\_perdhour\_per\_year(df\_s, ax, dt=1, smooth=True, color=f'C{ct}', alpha=0.8, lw=2, label=dow, ls='--', weight\_fun=wfun)

ax.set\_ylabel('Fraction of weekly emails per hour')

plt.legend(loc='upper left')



**OBSERVATION**: Order of received mails are

1.Tuesday

2.Thursday

3.Monday

4.Wednesday

5.Saturday

6.Sunday

7.Friday

from wordcloud import WordCloud

df\_no\_arxiv = dfs[dfs['from'] != 'no-reply@arXiv.org']

text = ' '.join(map(str, received['subject'].values))

from wordcloud import WordCloud

df\_no\_arxiv = dfs[dfs['from'] != 'no-reply@arXiv.org']

text = ' '.join(map(str, received['subject'].values))

stopwords = ['Re', 'Fwd', '3A\_']

wrd = WordCloud(width=700, height=480, margin=0, collocations=False)

for sw in stopwords:

 wrd.stopwords.add(sw)

wordcloud = wrd.generate(text)

plt.figure(figsize=(25,15))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.margins(x=0, y=0)

