Whats App Chat Analysis

App1.py file

import streamlit as st  
import preprocessor  
import helper  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
st.sidebar.title("Whatsapp Chat Analysis")  
  
uploaded\_file=st.sidebar.file\_uploader("Choose a File")  
if uploaded\_file is not None:  
 bytes\_data=uploaded\_file.getvalue()  
  
 # brings the data in string format from preprocessor  
 data=bytes\_data.decode("utf-8")  
 #st.text(data)  
  
 # now returning as dataframe  
 df=preprocessor.preprocess(data)  
 st.dataframe(df)  
  
 #fetching unique users  
 user\_list=df['user'].unique().tolist()  
 user\_list.remove("group\_notification")  
 user\_list.sort()  
 user\_list.insert(0,'Overall')  
 selected\_user=st.sidebar.selectbox("Show analysis with respect to",user\_list)  
  
 if st.sidebar.button("Show Analysis"):  
 num\_messages, words,num\_media\_messages,num\_links = helper.fetch\_stats(selected\_user, df)  
 st.title("Top Statistics of Whatsapp Chat")  
 col1,col2,col3,col4=st.columns(4)  
  
 with col1:  
 st.header("Total Messages")  
 st.title(num\_messages)  
  
 with col2:  
 st.header("Total Words")  
 st.title(words)  
  
 with col3:  
 st.header("Media Shared")  
 st.title(num\_media\_messages)  
  
 with col4:  
 st.header("Links Shared")  
 st.title(num\_links)  
  
  
 if selected\_user=='Overall':  
 st.title("Most Busy Users")  
 x,new\_df=helper.most\_busy\_users(df)  
 fig, ax= plt.subplots()  
  
 col1, col2=st.columns(2)  
  
 with col1:  
 ax.bar(x.index,x.values,color='teal')  
 st.pyplot(fig)  
  
 with col2:  
 st.dataframe(new\_df)  
  
 most\_common\_df = helper.most\_common\_words(selected\_user, df)  
 st.dataframe(most\_common\_df)  
  
 fig,ax=plt.subplots()  
  
 ax.bar(most\_common\_df[0],most\_common\_df[1],color='orange')  
 plt.xticks(rotation='vertical')  
 st.title("Most Common Words")  
 st.pyplot(fig)  
  
 emoji\_df=helper.emoji\_helper(selected\_user, df)  
 st.title("Emoji Analysis")  
  
 col1,col2=st.columns(2)  
  
 with col1:  
 st.dataframe(emoji\_df)  
 with col2:  
 fig,ax=plt.subplots()  
 ax.pie(emoji\_df[1].head(),labels=emoji\_df[0].head(),autopct="%0.2f")  
 st.pyplot(fig)  
  
#monthly timeline  
 st.title("Monthly Timeline Analyis")  
 timeline=helper.monthly\_timeline(selected\_user, df)  
 fig, ax = plt.subplots()  
 ax.plot(timeline['time'],timeline['message'])  
 plt.xticks(rotation='vertical')  
 st.pyplot(fig)  
  
#daily timeline  
 st.title("Daily Timeline Analyis")  
 daily\_timeline = helper.daily\_timeline(selected\_user, df)  
 fig, ax = plt.subplots()  
 ax.plot(daily\_timeline['only\_date'], daily\_timeline['message'],color='red')  
 plt.xticks(rotation='vertical')  
 st.pyplot(fig)  
  
  
 st.title("Activity Map")  
 col1,col2=st.columns(2)  
  
 with col1:  
 st.header("Most Busy Day")  
 busy\_day = helper.weekly\_activity\_map(selected\_user, df)  
 fig, ax = plt.subplots()  
 ax.bar(busy\_day.index, busy\_day.values, color='purple')  
 plt.xticks(rotation='vertical')  
 st.pyplot(fig)  
  
 with col2:  
 st.header("Most Busy Month")  
 busy\_month = helper.month\_activity\_map(selected\_user, df)  
 fig, ax = plt.subplots()  
 ax.bar(busy\_month.index, busy\_month.values, color='magenta')  
 plt.xticks(rotation='vertical')  
 st.pyplot(fig)  
  
 st.title("Online Activity Map")  
 user\_heatmap = helper.activity\_heatmap(selected\_user, df)  
 fig, ax = plt.subplots()  
 ax=sns.heatmap(user\_heatmap)  
 st.pyplot(fig)

Helper.py file

from urlextract import URLExtract  
import pandas as pd  
from collections import Counter  
import emoji  
import seaborn as sns  
  
extractor=URLExtract()  
def fetch\_stats(selected\_user,df):  
  
 if selected\_user !='Overall':  
 df=df[df['user'] == selected\_user]  
 # fetching no. of msgs  
  
 num\_messages=df.shape[0]  
 #no. of words  
  
 words=[]  
 for message in df['message']:  
 words.extend(message.split())  
  
 num\_media\_messages=df[df['message']=='<Media omitted>\n'].shape[0]  
  
 links=[]  
 for message in df['message']:  
 links.extend(extractor.find\_urls(message))  
  
 return num\_messages, len(words), num\_media\_messages, len(links)  
  
def most\_busy\_users(df):  
 x=df['user'].value\_counts().head()  
  
 df = round((df['user'].value\_counts() / df.shape[0]) \* 100, 2).reset\_index().rename(  
 columns={'index': 'name', 'user': 'percent'})  
 return x,df  
  
def most\_common\_words(selected\_user,df):  
 f=open('stop\_hinglish.txt','r')  
 stop\_words=f.read()  
  
 if selected\_user !='Overall':  
 df=df[df['user'] == selected\_user]  
  
 temp=df[df['user']!= 'group\_notification']  
 temp=temp[temp['message']!='<Media omitted>\n']  
  
 words=[]  
  
 for message in temp['message']:  
 for word in message.lower().split():  
 if word not in stop\_words:  
 words.append(word)  
  
 most\_common\_df=pd.DataFrame(Counter(words).most\_common(25))  
 return most\_common\_df  
  
def emoji\_helper(selected\_user,df):  
 if selected\_user !='Overall':  
 df=df[df['user'] == selected\_user]  
  
 emojis=[]  
  
 for message in df['message']:  
 emojis.extend([c for c in message if c in emoji.EMOJI\_DATA])  
  
 emoji\_df=pd.DataFrame(Counter(emojis).most\_common(len(Counter(emojis))))  
  
 return emoji\_df  
  
def monthly\_timeline(selected\_user,df):  
 if selected\_user !='Overall':  
 df=df[df['user'] == selected\_user]  
  
 timeline=df.groupby(['year','month\_num','month']).count()['message'].reset\_index()  
  
 time=[]  
 for i in range(timeline.shape[0]):  
 time.append(timeline['month'][i]+ '-' + str(timeline['year'][i]))  
  
 timeline['time']=time  
  
 return timeline  
  
def daily\_timeline(selected\_user,df):  
 if selected\_user !='Overall':  
 df=df[df['user'] == selected\_user]  
  
 daily\_timeline=df.groupby('only\_date').count()['message'].reset\_index()  
  
 return daily\_timeline  
  
def weekly\_activity\_map(selected\_user,df):  
 if selected\_user !='Overall':  
 df=df[df['user'] == selected\_user]  
  
 return df['day\_name'].value\_counts()  
  
def month\_activity\_map(selected\_user,df):  
 if selected\_user != 'Overall':  
 df = df[df['user'] == selected\_user]  
  
 return df['month'].value\_counts()  
  
def activity\_heatmap(selected\_user,df):  
 if selected\_user != 'Overall':  
 df = df[df['user'] == selected\_user]  
  
 user\_heatmap = df.pivot\_table(index='day\_name', columns='period', values='message', aggfunc='count').fillna(0)  
  
 return user\_heatmap

preprocessor.py file

import re  
import pandas as pd  
  
def preprocess(data):  
 pattern = '\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s-\s'  
  
 messages = re.split(pattern, data)[1:]  
 dates = re.findall(pattern, data)  
  
 df = pd.DataFrame({'user\_message': messages, 'message\_date': dates})  
 # convert message\_date type  
 df['message\_date'] = pd.to\_datetime(df['message\_date'], format='%d/%m/%y, %H:%M - ')  
  
 df.rename(columns={'message\_date': 'date'}, inplace=True)  
  
 users = []  
 messages = []  
 for message in df['user\_message']:  
 entry = re.split('([\w\W]+?):\s', message)  
 if entry[1:]: # user name  
 users.append(entry[1])  
 messages.append(" ".join(entry[2:]))  
 else:  
 users.append('group\_notification')  
 messages.append(entry[0])  
  
 df['user'] = users  
 df['message'] = messages  
 df.drop(columns=['user\_message'], inplace=True)  
  
 df['year'] = df['date'].dt.year  
 df['month\_num'] = df['date'].dt.month  
 df['only\_date'] = df['date'].dt.date  
 df['month'] = df['date'].dt.month\_name()  
 df['day'] = df['date'].dt.day  
 df['day\_name'] = df['date'].dt.day\_name()  
 df['hour'] = df['date'].dt.hour  
 df['minute'] = df['date'].dt.minute  
  
 period = []  
 for hour in df[['day\_name', 'hour']]['hour']:  
 if hour == 23:  
 period.append(str(hour) + "-" + str('00'))  
 elif hour == 0:  
 period.append(str('00') + "-" + str(hour + 1))  
 else:  
 period.append(str(hour) + "-" + str(hour + 1))  
  
 df['period'] = period  
  
 #previously returned as string , now it will return as dataframe  
 return df