This notebook uses BERTweet to compute the daily sentiment of non-commercial, non-spam accounts in the MIT Vape and JUUL tweet Dataset from 2015 to 2022

Each tweet is weighted by 1/(the number of tweets the user made that day), and the daily proportion of Negative, Neutral, and Positive sentiment are then computed and plotted by date

Pulling dataset from Kaggle

```
import zipfile
import os
from google.colab import files

files.upload()

!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d bestgirl/vape-and-juul-tweet-dataset


with zipfile.ZipFile('vape-and-juul-tweet-dataset.zip', 'r') as zip_ref:
    zip_ref.extractall('vape_and_juul')

files = []
for dirname, _, filenames in os.walk('vape_and_juul'):
    for filename in filenames:
        files.append(os.path.join(dirname, filename))

```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 64 bytes, last modified: 8/28/2025 - 100% done

```
Saving kaggle.json to kaggle (2).json
mkdir: cannot create directory '/root/.kaggle': File exists
Dataset URL: https://www.kaggle.com/datasets/bestgirl/vape-and-juul-tweet-dataset
License(s): MIT
vape-and-juul-tweet-dataset.zip: Skipping, found more recently modified local copy (use --force to force download)
```

Below is a duplicate file being removed and justification for why it is a duplicate

```
xlsx_files = [f for f in files if f.lower().endswith(".xlsx")]
xlsx_files # this caused the program to crash earlier, I have a csv version in there as well so I am removing it from files since it is
```

```
['vape_and_juul/2020-08/OTS-2020-08-04.xlsx']
```

```
for f in files:
    if f[-14:-4] == '2020-08-04':
        print(f)
```

```
vape_and_juul/2020-08/OTS-2020-08-04.csv
```

```
# removing the duplicate xlsx
files.remove(xlsx_files[0])
```

Only selecting OTS files

```
OTS_only = []
for file in files:
  if file[22:25] =='OTS':
    OTS_only.append(file)
files = OTS_only

```

Instantiating BERTweet

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline

model_name = "finiteautomata/bertweet-base-sentiment-analysis"

tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast = True)
BERTweet = AutoModelForSequenceClassification.from_pretrained(model_name)

sentiment_pipeline = pipeline(
    "sentiment-analysis",
    model = BERTweet,
    tokenizer = tokenizer,
    device = 0
)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
emoji is not installed, thus not converting emoticons or emojis into text. Install emoji: pip3 install emoji==0.6.0
Device set to use cuda:0
```

Computing Sentiment Proportions by Date

```
from tqdm import tqdm
import pandas as pd

tweets_today = []
dates = []
NEGS = []
NEUS = []
POSS = []
files_with_issues = []
empty_after_spam = []
for file in tqdm(files, desc = "Processing files"):

    try:
        # Taking in new file and creating df
        df = pd.read_csv(file, on_bad_lines='skip',engine = 'python', encoding = 'utf-8')
    except Exception:
        print(f"there is an error with file: {file}, do it by itself down below")
        files_with_issues.append(file)
        continue
    df = df.dropna()

    # Removing potential retweets and stores
    df = df[~df['TweetContent'].str.contains("Repost|RT|% off|buy|coupon|sale|discount|free|shop|store|deal|get_repost|get yours today|g

    # Making a dictionary of tweet counts and usernames
    tweet_counts = df.groupby('TweeterUsername').size()
    tweet_counts = tweet_counts.to_dict()

    # Spam handling
    # Since I dont have a bot detecter, I am going to set a hard rule that more than 20 posts in a day is spam and they are either a bot
    spam_users = set()
    for k, v in tweet_counts.items():
        if v > 20:
            spam_users.add(k)

    df = df[~df['TweeterUsername'].isin(spam_users)]

    # Some more spam handling / ensuring no issues with BERTweet token capacity

    if df['TweetContent'].tolist() == []:
        # this happend with one of them. Not sure why but will check out in the individual basis ones
        empty_after_spam.append(file)

        continue

    encodings = tokenizer(df['TweetContent'].tolist(), add_special_tokens = True, truncation = False, padding = False)
```

```python
    df['token_length'] = [len(ids) for ids in encodings['input_ids']]

    df = df[df['token_length'] <=128]


  # Running BERTweet on all the remaining valid tweets
    input = df['TweetContent'].to_list()
    results = sentiment_pipeline(input, batch_size = 128)


    # Weighting sentiment per post based on number of posts made. Each user gets an allocation of 1 unit
    sentiment_weight = {}
    for k, v in tweet_counts.items():
        if k not in spam_users:
            sentiment_weight[k] = float(1/v)

    # Adding Sentiment and Sentiment weight to the df, the weight is determined by how many total posts they made
    df['Sentiment'] = [r['label'] for r in results]
    df['Sentiment_weight'] = df['TweeterUsername'].map(sentiment_weight)


    # Computing daily tweet count weighted proportions
    t = len(set(df['TweeterUsername']))

    df_POS = df[df['Sentiment'] == 'POS']
    POS_prop = df_POS['Sentiment_weight'].sum() / t

    df_NEU = df[df['Sentiment'] == 'NEU']
    NEU_prop = df_NEU['Sentiment_weight'].sum() / t

    df_NEG = df[df['Sentiment'] == 'NEG']
    NEG_prop = df_NEG['Sentiment_weight'].sum() / t

    # Adding Proportions to respective lists
    NEGS.append(NEG_prop)
    NEUS.append(NEU_prop)
    POSS.append(POS_prop)
    tweets_today.append(len(df))

    # Adding the date to dates
    date = pd.to_datetime(file[-14:-4])
    dates.append(date)


```

```
Processing files: 100%|██████████| 2891/2891 [2:31:46<00:00,  3.15s/it]
```

```python
files_with_issues
```

```
[]
```

```python

empty_after_spam
```

```
['vape_and_juul/2022-11/OTS-2022-11-29.csv',
 'vape_and_juul/2022-11/OTS-2022-11-30.csv']
```

Creating the Data Frame

```python
master_df = pd.DataFrame({
    'date' : dates,
    'NEG_prop': NEGS,
    'NEU_prop': NEUS,
    'POS_prop': POSS,
    'tweets_today': tweets_today
})
master_df['date'] = pd.to_datetime(master_df['date'])
master_df = master_df.sort_values(by = 'date')
master_df
```

```
         date  NEG_prop  NEU_prop  POS_prop  tweets_today
520   2015-01-01  0.139855  0.437129  0.423016          1187
503   2015-01-02  0.143684  0.500984  0.355332          1374
506   2015-01-03  0.202525  0.455398  0.342078          1346
498   2015-01-04  0.184096  0.472082  0.343821          1354
497   2015-01-05  0.183777  0.480495  0.335728          1652
...          ...       ...       ...       ...           ...
2254  2022-11-24  0.402320  0.417159  0.180493          2522
2265  2022-11-25  0.383537  0.422500  0.193963          2220
2274  2022-11-26  0.443789  0.403812  0.152399          2208
2275  2022-11-27  0.414121  0.436821  0.149058          2064
2280  2022-11-28  0.395215  0.440867  0.162505           408

2889 rows × 5 columns
```

Next steps: Generate code with master_df | View recommended plots | New interactive sheet

Saving master_df to a file in case anything goes wrong:

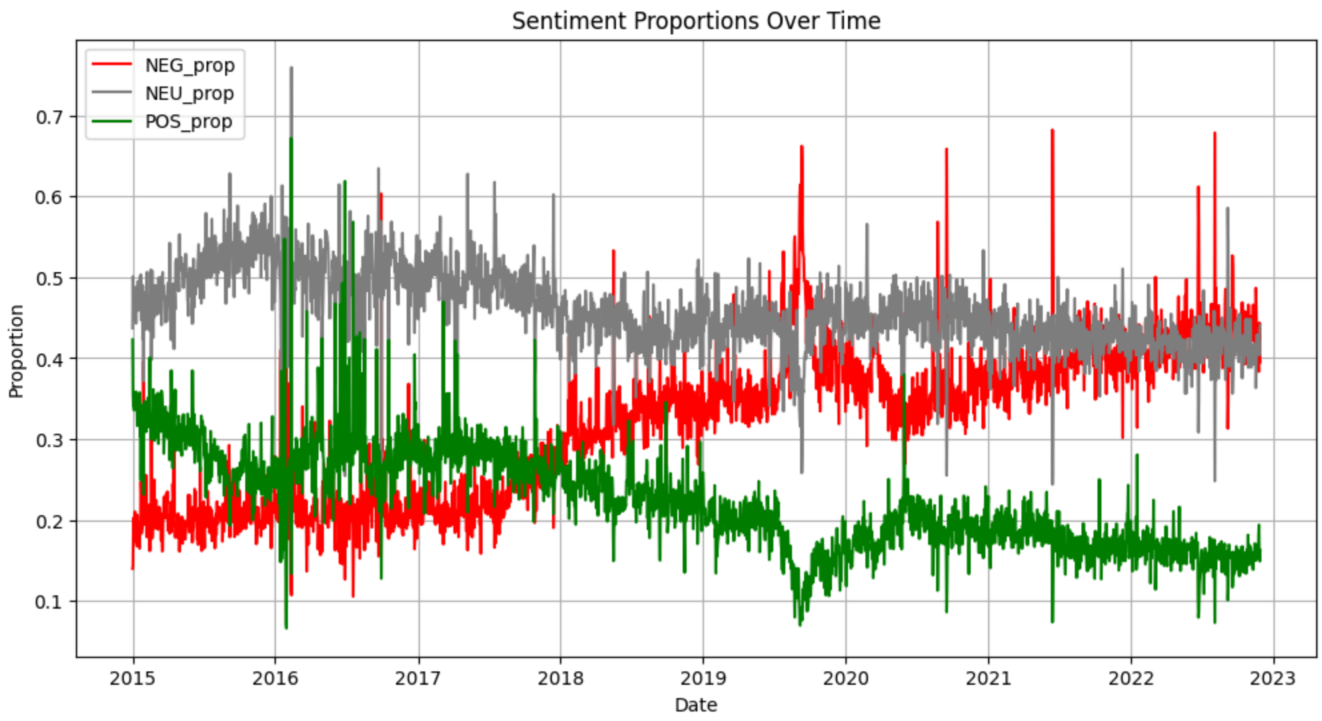
```
from google.colab import files
master_df.to_csv("MIT_Vape_and_JUUL_sentiment.csv", index=False)
files.download("MIT_Vape_and_JUUL_sentiment.csv")
```

Inspecting: 'vape_and_juul/2022-11/OTS-2022-11-29.csv', and 'vape_and_juul/2022-11/OTS-2022-11-30.csv'

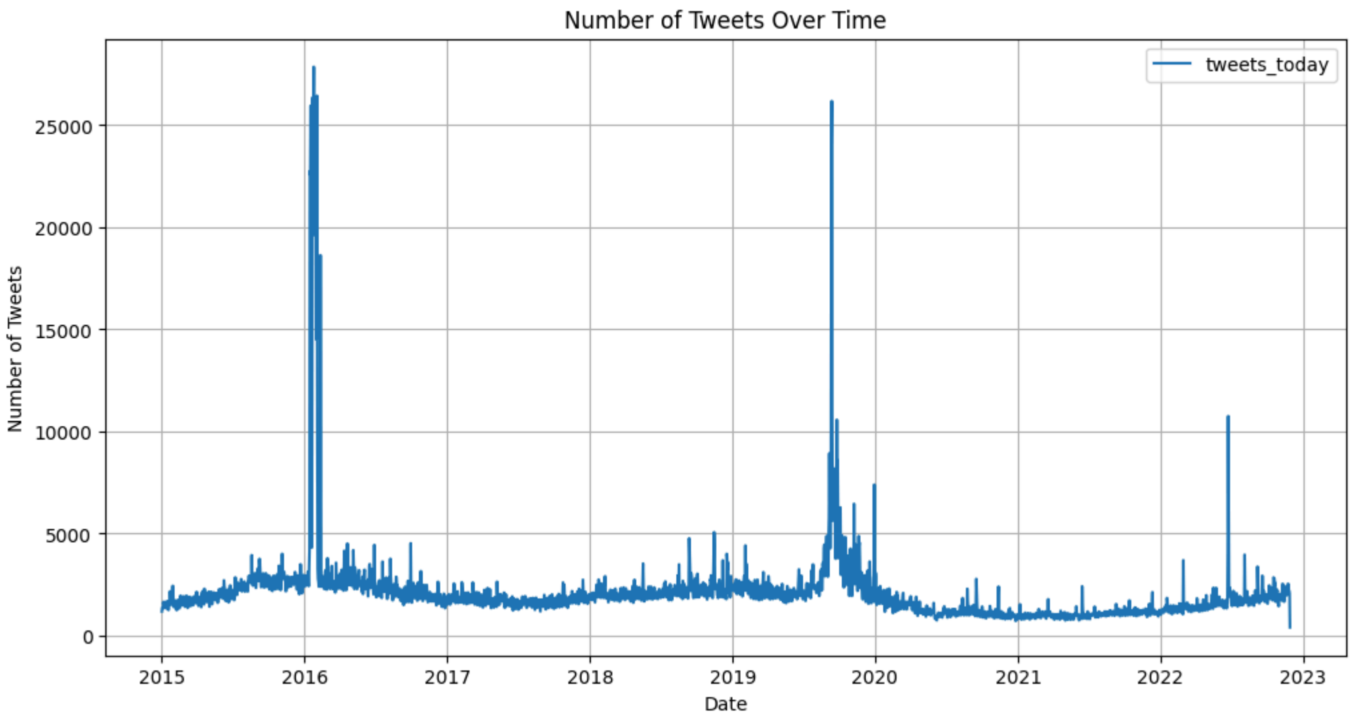
The actual files are empty and these are the very last date so they will just be ommited from the set.

Plotting Sentiment Proportions

```
import matplotlib.pyplot as plt


plt.figure(figsize=(12, 6))

plt.plot(master_df['date'], master_df['NEG_prop'], label='NEG_prop', color = 'red')
plt.plot(master_df['date'], master_df['NEU_prop'], label='NEU_prop',  color = 'grey')
plt.plot(master_df['date'], master_df['POS_prop'], label='POS_prop',  color = 'green')

plt.title("Sentiment Proportions Over Time")
plt.xlabel("Date")
plt.ylabel("Proportion")
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 6))

plt.plot(master_df['date'], master_df['tweets_today'], label='tweets_today')

plt.title("Number of Tweets Over Time")
plt.xlabel("Date")
plt.ylabel("Number of Tweets")
plt.legend()
plt.grid(True)
plt.show()
```



Sentiment Proportions Over Time
NEG_prop
NEU_prop
POS_prop
Proportion
Date
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2015
2016
2017
2018
2019
2020
2021
2022
2023



Number of Tweets Over Time
tweets_today
Number of Tweets
Date
0
5000
10000
15000
20000
25000
2015
2016
2017
2018
2019
2020
2021
2022
2023