TWITTER CLIMATE CHANGE SENTIMENT ANALYSIS

PROJECT TEAM MEMBERS

1. Eric Cheruiyot
2. Geoffrey Ongeri
3. Antonia Mulinge

INTRODUCTION

A changing climate is leading to more occurrences of extreme events such as droughts (moisture deficits) and floods (moisture surpluses), which have a negative impact on crop growth and yields. Plants require an optimum soil-water-air environment in the root zone to maintain physiological response to growth, photosynthetic functions, and productive vegetative capacity for high yield response.

Rising air temperatures also have a deleterious effect on crop production, as heat stress limits the optimum productive capability of our current commercial plant species on which the population depends for food security.

A changing climate is also affecting other aspects of crop production such as increased pests and disease in some situations, changes and even losses in biodiversity in some environments, and major disruptions due to extreme storm events, such as the recent decimation by Hurricane Maria of Puerto Rico's agricultural crops, livestock, and associated infrastructure.

Problem Statement

Twitter being one of the most influential ways of passing information and analysing trends on various issues and in our case climate change, we chose to analyse tweets data to predict whether a tweet is pro-climate change, anti-climate change , from a news segment or just a neutral part with not take in climate change issues.

Metric for success

Implementing a model that can accurately predict whether a tweet lies in the following categories of sentiments:

- Pro
- Anti
- News
- Neutral

These categories are all in relation to climate change.

Data Source: Kaggle

Context

The dataset in its entirity is an extract from twitter and involves the message of climate change. In recent times there has been alot of talk aroud climate change and preseving our climate.

Data Relevance

The collection of this data was funded by a Canada Foundation for Innovation JELF Grant to Chris Bauch, University of Waterloo.

This dataset aggregates tweets pertaining to climate change collected between Apr 27, 2015 and Feb 21, 2018. In total, 43943 tweets were annotated. Each tweet is labelled independently by 3 reviewers. This dataset only contains tweets that all 3 reviewers agreed on (the rest were discarded).

Data set description

Each tweet is labelled as one of the following classes:

- *2(News)*: the tweet links to factual news about climate change
- *1(Pro)*: the tweet supports the belief of man-made climate change
- *0(Neutral)*: the tweet neither supports nor refutes the belief of man-made climate change
- *-1(Anti)*: the tweet does not believe in man-made climate change

Table of contents

Refresh

EXPLORATORY DATA ANALYSIS

```
#Importing and data manipulation libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
%matplotlib inline
import re
import string

#Text preprocessing libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

1. **Loading the dataset into the notebook and proceeding to visualize the first 5 rows of the dataset**

```
#Loading and previewing the data
df = pd.read_csv('twitter_sentiment_data.csv')
df.head()
```

	sentiment	message	tweetid
0	-1	@tiniebeany climate change is an interesting h...	792927353886371840
1	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...	793124211518832641
2	1	Fabulous! Leonardo #DiCaprio's film on #climat...	793124402388832256
3	1	RT @Mick_Fanning: Just watched this amazing do...	793124635873275904
4	2	RT @cnalive: Pranita Biswasi, a Lutheran from ...	793125156185137153

2. Finding out the shape of the dataset

```
#check shape
df.shape
```

```
(43943, 3)
```

The dataset contains 43943 rows and 3 columns

The rows represent tweets and each tweet has 3 features

3. **Checking data types per column**

```
#check data types
df.dtypes
```

```
sentiment     int64
```

```
message      object
tweetid       int64
dtype: object
```

The sentiment and tweetid columns are of int type, while message is a string, however tweeetid should be converted to string wheres sentiment should be categorical

4. Getting overview of the data

```
#check data overview
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43943 entries, 0 to 43942
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   sentiment  43943 non-null  int64 
 1   message    43943 non-null  object
 2   tweetid    43943 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 1.0+ MB
```

4. a). Searching for null/missing columns

```
#check for null values
df.isnull().values.any()
#Returns false meaning there are no missing values/null values
```

```
False
```

Returns false meaning there are no missing values/null values

6. Checking column titles

```
#check column titles
df.columns
```

```
Index(['sentiment', 'message', 'tweetid'], dtype='object')
```

As stated earlier, there are 3 columns namely:

- sentiment

- message
- tweetid

7. Dropping the tweetid column and previewing the changes have been effected.

```
#drop tweetid column
df.drop('tweetid', axis=1, inplace=True)
df.head(10)
```

	sentiment	message
0	-1	@tiniebeany climate change is an interesting h...
1	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...
2	1	Fabulous! Leonardo #DiCaprio's film on #climat...
3	1	RT @Mick_Fanning: Just watched this amazing do...
4	2	RT @cnalive: Pranita Biswasi, a Lutheran from ...
5	0	Unamshow awache kujinga na iko global warming ...
6	2	RT @cnalive: Pranita Biswasi, a Lutheran from ...
7	2	RT @CCIRiviera: Presidential Candidate #Donald...
8	0	RT @AmericanIndian8: Leonardo DiCaprio's clima...
9	1	#BeforeTheFlood Watch #BeforeTheFlood right he...

```
#check our data
df
```

	sentiment	message
0	-1	@tiniebeany climate change is an interesting h...
1	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...
2	1	Fabulous! Leonardo #DiCaprio's film on #climat...

```
#check our data for duplicate rows total
df. duplicated(). sum()
```

```
2902
```

... ...

```
#check our data for duplicate rows total
duplicate_rows = df[df.duplicated()]
duplicate_rows
```

	sentiment	message
6	2	RT @cnalive: Pranita Biswasi, a Lutheran from ...
12	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...
19	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...
26	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...
29	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...
...	...	...
40535	1	RT @antiarzE: - do u like green eggs &amp; ham...
42668	0	RT @exostext: Bbh: boys are hot\nBbh: girls ar...
42697	-1	RT @realDonaldTrump: The global warming we sho...
43132	1	RT @ClimateReality: We can’t fight climate cha...
43825	1	RT @billmaher: Not a single question about cli...

2902 rows × 2 columns

```
#drop duplicate rows total
df.drop_duplicates()
```

	sentiment	message
0	-1	@tiniebeany climate change is an interesting h...
1	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...
2	1	Fabulous! Leonardo #DiCaprio's film on #climat...
3	1	RT @Mick_Fanning: Just watched this amazing do...
4	2	RT @cnalive: Pranita Biswasi, a Lutheran from ...
...	...	...
43938	1	Dear @realDonaldTrump,\nYeah right. Human Medi...
43940	2	RT @MikkiL: UN Poll Shows Climate Change Is th

Converting the sentiment column to names in the categories

```
#Converting the sentiment column to names in the categories
sentiment_num2name = {
    -1: "Anti",
     0: "Neutral",
     1: "Pro",
     2: "News",
}
df["sentiment"] = df["sentiment"].apply(lambda num: sentiment_num2name[num])
df.head()
```

	sentiment	message
0	Anti	@tiniebeany climate change is an interesting h...
1	Pro	RT @NatGeoChannel: Watch #BeforeTheFlood right...
2	Pro	Fabulous! Leonardo #DiCaprio's film on #climat...
3	Pro	RT @Mick_Fanning: Just watched this amazing do...
4	News	RT @cnalive: Pranita Biswasi, a Lutheran from ...

9. Investigating the distribution of Sentiments from -1 to 2

```
#Checking the class distribution
df['sentiment'].value_counts()
```

```
Pro        22962
News        9276
Neutral     7715
```

```
Anti        3990
Name: sentiment, dtype: int64
```

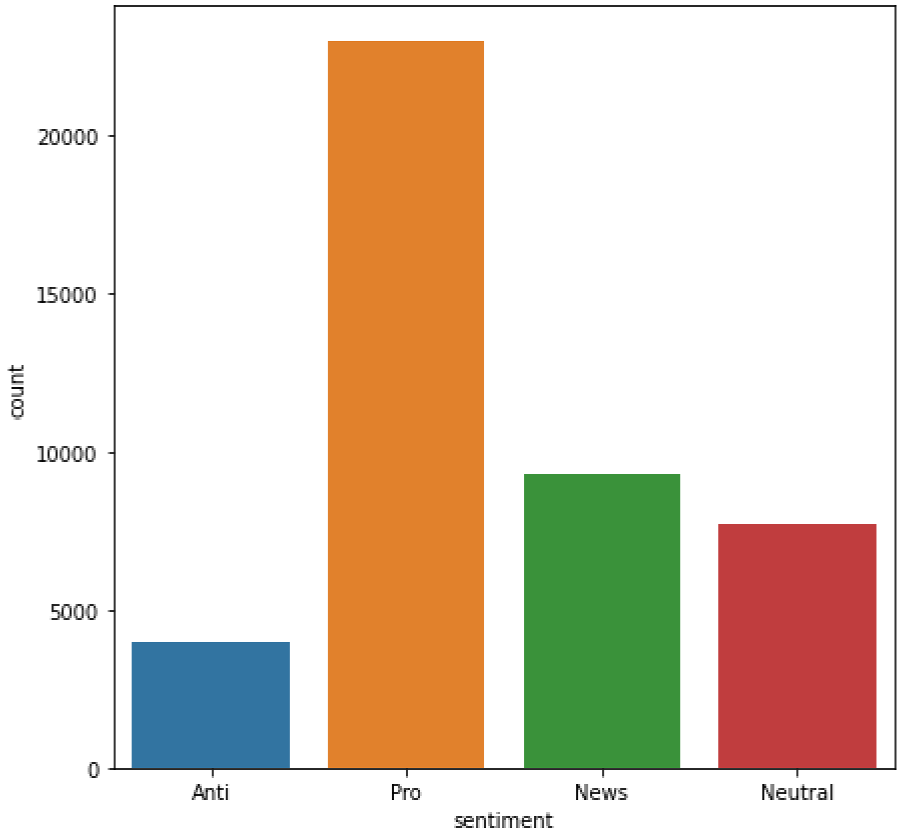
From the result above, the categories are as follows:

- 2 = 9276 tweets
- 1 = 22962 tweets
- 0 = 7715 tweets
- -1 = 3990 tweets

Visualizing the sentiments distribution

```
#plot sentiments dist
fig = plt.figure(figsize=(7,7))
sns.countplot(x='sentiment', data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe4f9f6c950>
```

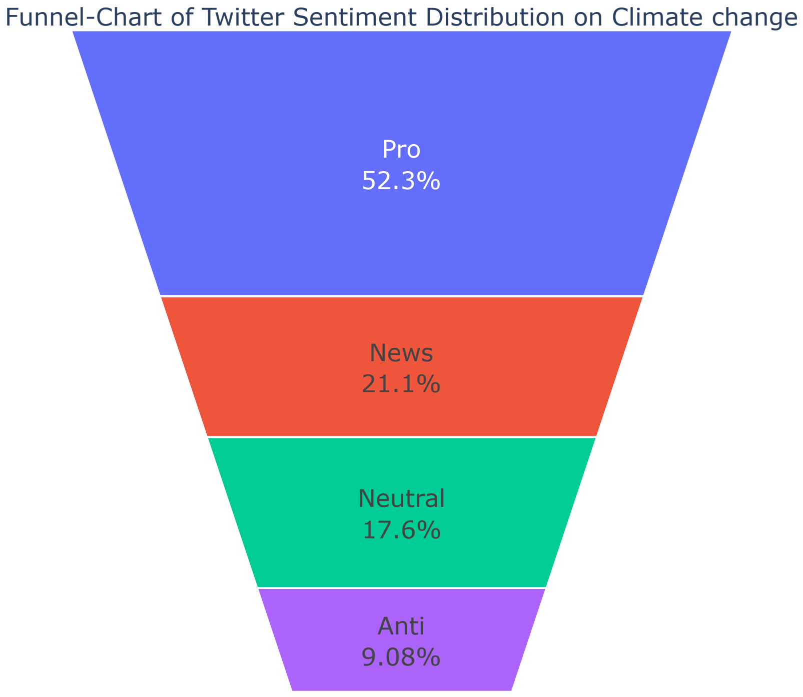


```
#table showing sum of sentiments
temp = df.groupby('sentiment').count()['message'].reset_index().sort_values(by='message',asce
temp.style.background_gradient(cmap='Purples')
```

	sentiment	message
3	Pro	22962
2	News	9276

```
#install plotly
!pip install plotly
from plotly import graph_objs as go

fig = go.Figure(go.Funnelarea(
    text =temp.sentiment,
    values = temp.message,
    title = {"position": "top center", "text": "Funnel-Chart of Twitter Sentiment Distributio
    ))
fig.show()
```

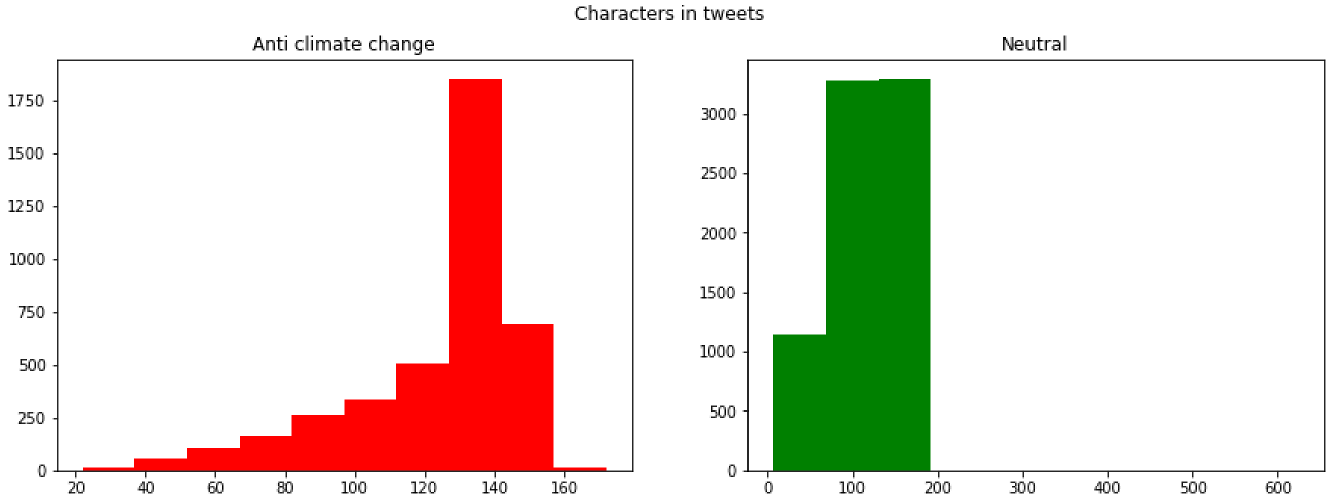
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (5.5.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from plot
```



Conclusively, from the visualizations above, it is clear that Category 1 contains the highest number of tweets which suggests that most tweets are pro- climate change. This is closely followed by category 2, tweets which are new or from news outlets and taking the last category, -1, are tweets which are anti-climate change

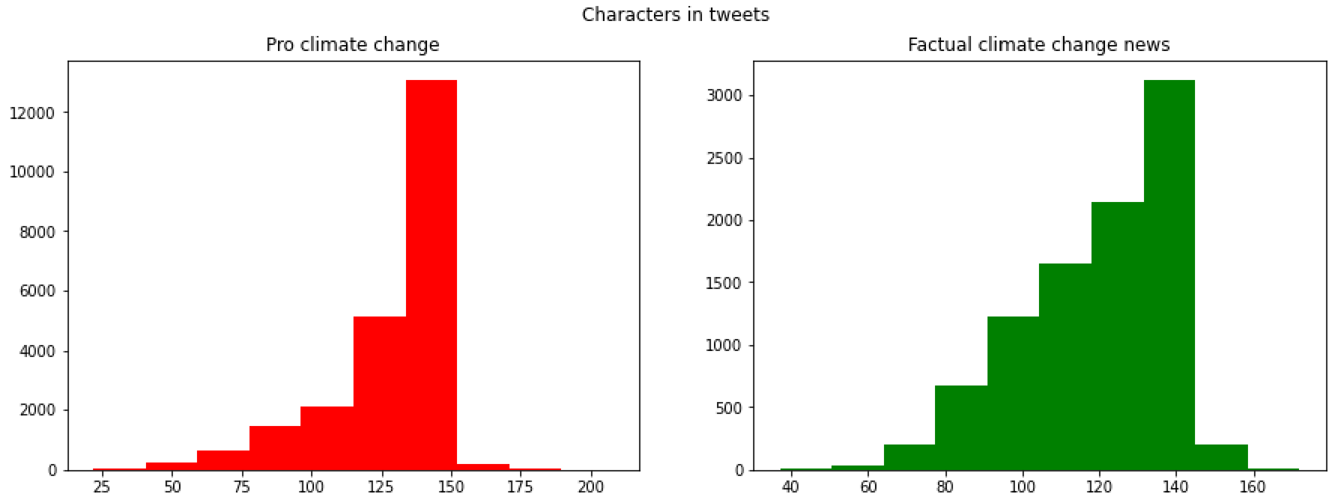
10. Investigating the number of characters, words in the tweets

```
#EDA
#Number of characters in tweet
fig,(ax1,ax2)=plt.subplots(1,2, figsize=(15,5))
tweet_len=df[df['sentiment']=='Anti']['message'].str.len()
ax1.hist(tweet_len,color='red')
ax1.set_title('Anti climate change')
tweet_len=df[df['sentiment']=='Neutral']['message'].str.len()
ax2.hist(tweet_len,color='green')
ax2.set_title('Neutral')
fig.suptitle('Characters in tweets')
plt.show()
```



```
fig,(ax1,ax2)=plt.subplots(1,2, figsize=(15,5))
tweet_len=df[df['sentiment']=='Pro']['message'].str.len()
ax1.hist(tweet_len,color='red')
ax1.set_title('Pro climate change')
tweet_len=df[df['sentiment']=='News']['message'].str.len()
ax2.hist(tweet_len,color='green')
ax2.set_title('Factual climate change news')
```

```
fig.suptitle('Characters in tweets')
plt.show()
```

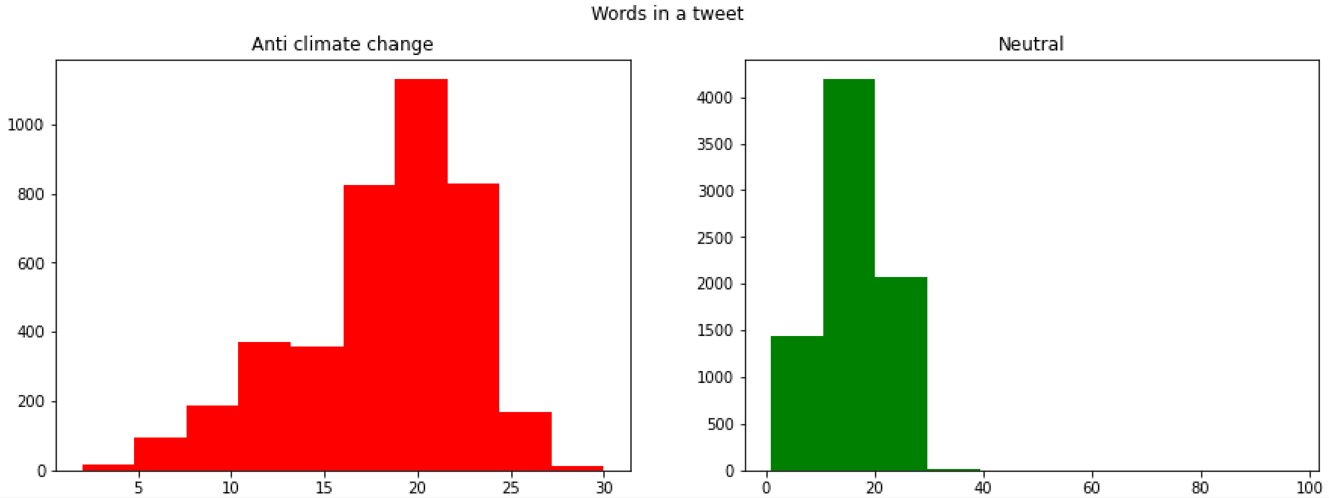


From the visualizations above, the characters in a tweet range from 0 to about 180 for anti-climate change, while those of neutral tweets range between 0 to about 180 characters, pro-climate change tweets have about 0 to about 180 characters while those of factual climate change news range from 0 to about 180 characters

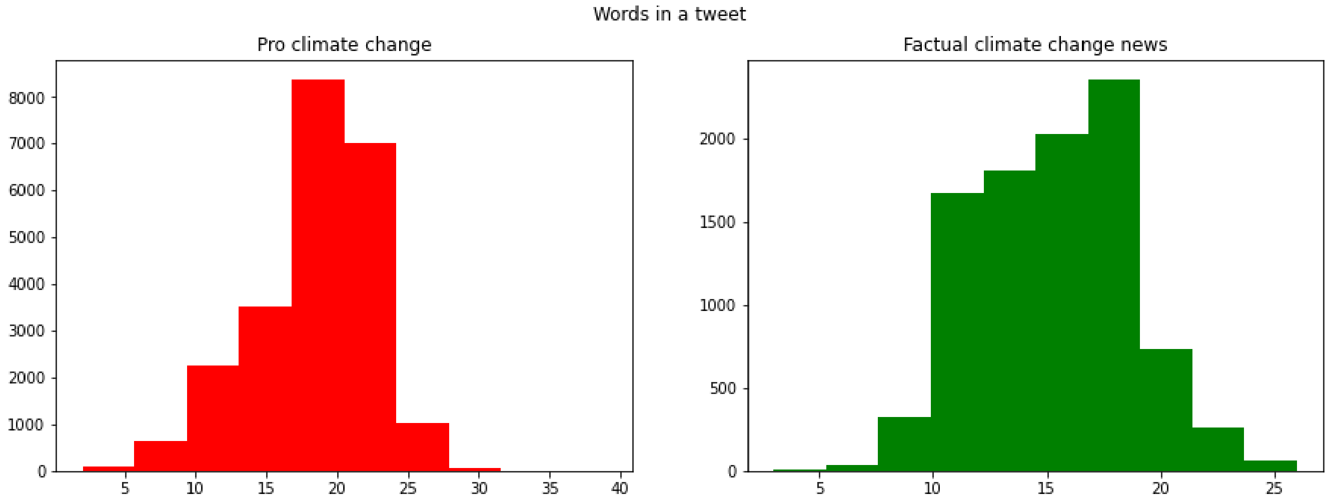
---

10. b). Investigating words in tweets

```
#Words in tweets

fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,5))
tweet_len=df[df['sentiment']=='Anti']['message'].str.split().map(lambda x: len(x))
ax1.hist(tweet_len,color='red')
ax1.set_title('Anti climate change')
tweet_len=df[df['sentiment']=='Neutral']['message'].str.split().map(lambda x: len(x))
ax2.hist(tweet_len,color='green')
ax2.set_title('Neutral')
fig.suptitle('Words in a tweet')
plt.show()
```



```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,5))
tweet_len=df[df['sentiment']=='Pro']['message'].str.split().map(lambda x: len(x))
ax1.hist(tweet_len,color='red')
ax1.set_title('Pro climate change')
tweet_len=df[df['sentiment']=='News']['message'].str.split().map(lambda x: len(x))
ax2.hist(tweet_len,color='green')
ax2.set_title('Factual climate change news')
fig.suptitle('Words in a tweet')
plt.show()
```

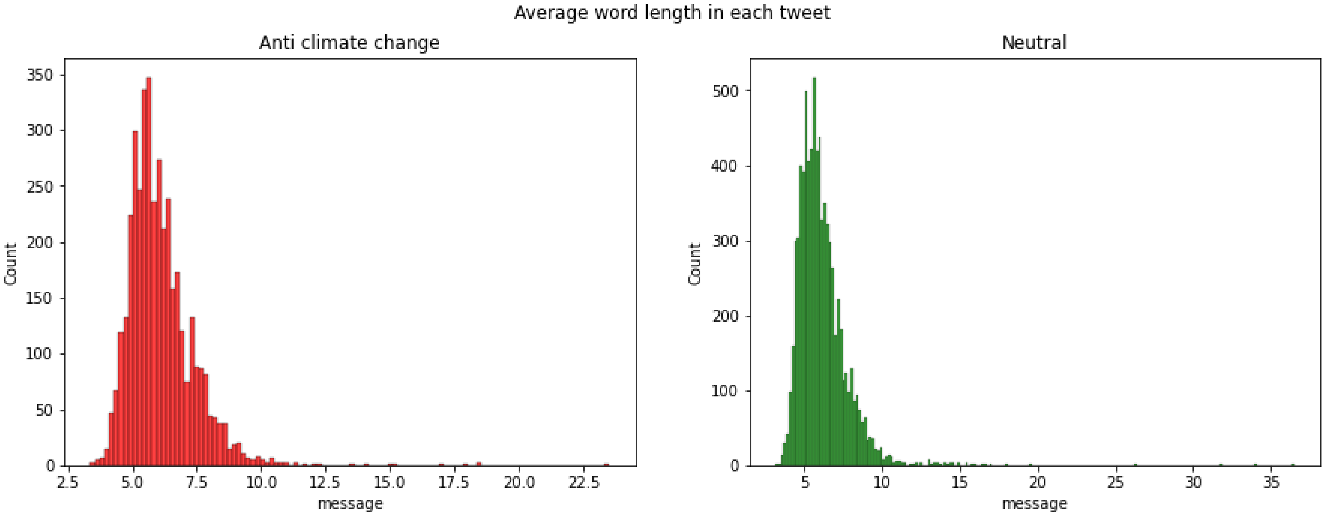


From the visualizations above, the words in a tweet range from 0 to about 30 for anti-climate change, while those of neutral tweets range between 0 to about 40 characters, pro-climate change tweets have about 0 to about 32 characters while those of factual climate change news range from 0 to about 25 characters

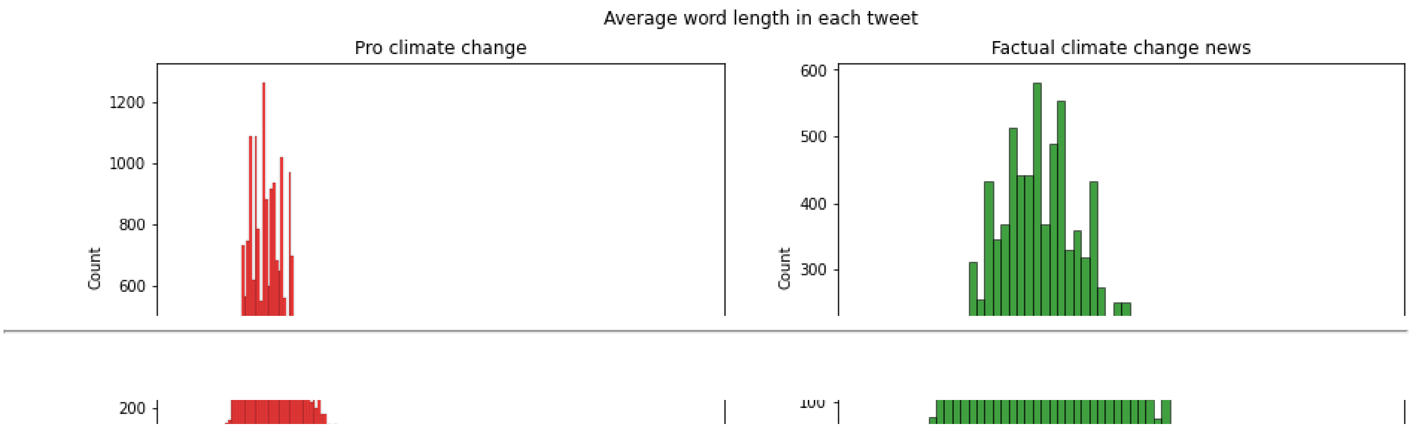
```
#Average word length
```

```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,5))
word=df[df['sentiment']=='Anti']['message'].str.split().apply(lambda x : [len(i) for i in x])
sns.histplot(word.map(lambda x: np.mean(x)),ax=ax1,color='red')
ax1.set_title('Anti climate change')
word=df[df['sentiment']=='Neutral']['message'].str.split().apply(lambda x : [len(i) for i in
sns.histplot(word.map(lambda x: np.mean(x)),ax=ax2,color='green')
ax2.set_title('Neutral')
fig.suptitle('Average word length in each tweet')
```

```
Text(0.5, 0.98, 'Average word length in each tweet')
```



```
#Average word length
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,5))
word=df[df['sentiment']=='Pro']['message'].str.split().apply(lambda x : [len(i) for i in x])
sns.histplot(word.map(lambda x: np.mean(x)),ax=ax1,color='red')
ax1.set_title('Pro climate change')
word=df[df['sentiment']=='News']['message'].str.split().apply(lambda x : [len(i) for i in x])
sns.histplot(word.map(lambda x: np.mean(x)),ax=ax2,color='green')
ax2.set_title('Factual climate change news')
fig.suptitle('Average word length in each tweet');
```



Tweet hashtag summary

Since Twitter uses Hashtags almost like a summarization feature (at least in the sense of highlighting core ideas). We look at some of top hashtags for each of the classes of sentiment. We'll then make "word clouds" to visualize their prominence.

```
#importing necessary libraries
import re
import nltk
import itertools
```

```
from IPython.display import (
    Markdown as md,
    Latex,
    HTML,
)
from tqdm.auto import tqdm
```

```
top15 = {}

by_sentiment = df.groupby("sentiment")
for sentiment, group in tqdm(by_sentiment):
    hashtags = group["message"].apply(lambda tweet: re.findall(r"#(\w+)", tweet))
    hashtags = itertools.chain(*hashtags)
    hashtags = [ht.lower() for ht in hashtags]

    frequency = nltk.FreqDist(hashtags)

    df_hashtags = pd.DataFrame({
        "hashtags": list(frequency.keys()),
        "counts": list(frequency.values()),
    })
    top15_htags = df_hashtags.nlargest(15, columns=["counts"])

    top15[sentiment] = top15_htags.reset_index(drop=True)
```

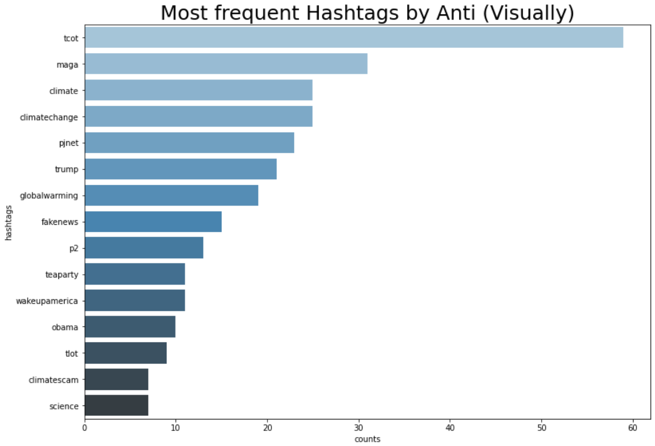
```
display(pd.concat(top15, axis=1).head(n=10))
```

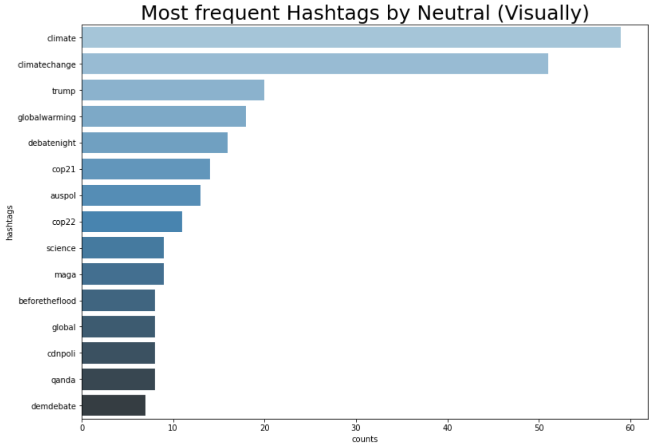
```
100%                                                  4/4 [00:00<00:00, 29.96it/s]
```

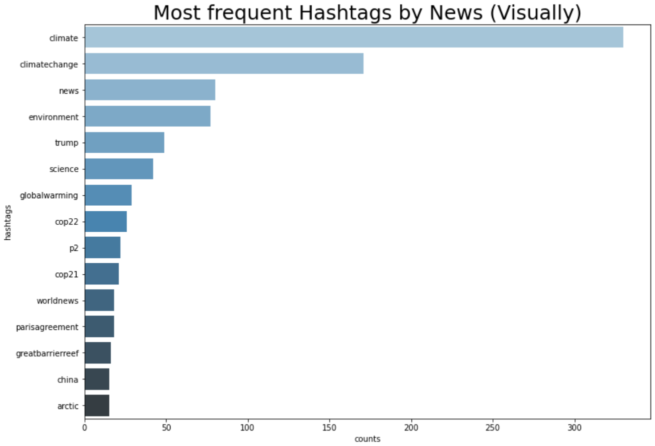
	Anti		Neutral		News		Pro	
	hashtags	**counts**	**hashtags**	**counts**	**hashtags**	**counts**	**hashtags**	**cou**
0	tcot	59	climate	59	climate	330	climate	
1	maga	31	climatechange	51	climatechange	171	climatechange	
2	climate	25	trump	20	news	80	beforetheflood	
3	climatechange	25	globalwarming	18	environment	77	actonclimate	
4	pjnet	23	debatenight	16	trump	49	parisagreement	
5	trump	21	cop21	14	science	42	imvotingbecause	
6	globalwarming	19	auspol	13	globalwarming	29	cop22	
7	fakenews	15	cop22	11	cop22	26	auspol	
8	p2	13	science	9	p2	22	cop21	
9	teaparty	11	maga	9	cop21	21	globalwarming	

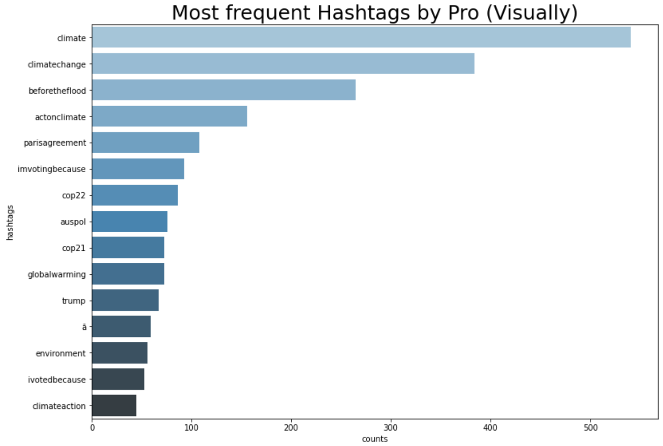
Visualizing the frequency of hashtags in barplots for each sentiment categorization

```
fig, axes = plt.subplots(2, 2, figsize=(28, 20))
counter = 0

for sentiment, top in top15.items():
    sns.barplot(data=top, y="hashtags", x="counts", palette="Blues_d", ax=axes[counter // 2,
    axes[counter // 2, counter % 2].set_title(f"Most frequent Hashtags by {sentiment} (Visual
    counter += 1
plt.show()
```









Observations:

- The most popular hashtags are, broadly, climate and climatechange. Which is expected, given the topic; but also, among the top 3 are relating to trump and his campaign slogan maga.
- The BeforeTheFlood hashtag refers to a 2016 documentary where Leonardo DiCaprio met with scientists, activists, and word leaders to discuss the dangers of climate and and possible solutions.
- COP22, ParisAgreement, and Trump in the Pro sentiment are likely related to the formal process Trump's administration began to exit the Paris Agreements, where north of 200 nations pledged to reduce greenhour gas emissions, assist developing nations, and assist [poor] nations struggling with the consequences of a warming Earth.
- Interestingly, auspol (short for Australian Politics) made the shortlist of the Pro sentiment. This is likely attributed to an assessment published quantifying the role of climate change in Australian brushfires and their increaseed risk of occuring.

CLEANING TWEET DATA

Tweet Data Cleaning

```
#Data pre-processing
#Glance at data
df['message'][:20]
```

```
0     @tiniebeany climate change is an interesting h...
1     RT @NatGeoChannel: Watch #BeforeTheFlood right...
2     Fabulous! Leonardo #DiCaprio's film on #climat...
3     RT @Mick_Fanning: Just watched this amazing do...
4     RT @cnalive: Pranita Biswasi, a Lutheran from ...
5     Unamshow awache kujinga na iko global warming ...
6     RT @cnalive: Pranita Biswasi, a Lutheran from ...
7     RT @CCIRiviera: Presidential Candidate #Donald...
8     RT @AmericanIndian8: Leonardo DiCaprio's clima...
9     #BeforeTheFlood Watch #BeforeTheFlood right he...
10    RT @DrDeJarnett: It's vital that the public he...
11    Bangladesh did not cause climate change, so th...
12    RT @NatGeoChannel: Watch #BeforeTheFlood right...
13    Now's the time: we need a strong #FTT that wor...
14    RT @WorldResources: Reflections on Leonardo Di...
15    RT @littoralsociety: The problem with global w...
16    RT @NatGeoChannel: One of the easiest ways to ...
17    RT @esquire: Watch Leo DiCaprio's climate chan...
18    RT @ClimateCentral: Here's how climate change ...
19    RT @NatGeoChannel: Watch #BeforeTheFlood right...
Name: message, dtype: object
```

defining the cleaning function to remove specified elements from the tweet message

```
#Define a function

import re

def text_cleaning(message):

    message = message.lower()                            #Lowercase
    message = re.sub(r'http\S+|www\.\S+|https\S+', '', message, flags=re.MULTILINE) #Removing
    message = re.sub(r'\@\w+|\#','', message)     # removing @mentions and #
    message = re.sub(r'rt[\s]+', '', message)           # removing RT
    message = re.sub(r'&[a-z;]+', '', message)          # removing '&gt;'
    message = re.sub('[^a-zA-Z]', ' ', message)         #removing punctuation

    return message
```

applying cleaning function and previewing changes

```
# applying the text cleaning function on tweets
df['message'] = df['message'].apply(text_cleaning)
df.head(100)
```

	sentiment	message
0	Anti	climate change is an interesting hustle as it...
1	Pro	watch beforetheflood right here as travels...
2	Pro	fabulous leonardo dicaprio s film on climate ...
3	Pro	just watched this amazing documentary by leo...
4	News	pranita biswasi a lutheran from odisha giv...
...	...	...
95	News	alberta tories are losing a big issue they ve ...
96	Pro	if you don t believe in global warming come ...
97	Pro	we just entered an alarming new era of glo...
98	Pro	global warming real as hell al gore told us...
99	News	listen to years of climate change in one mi...

100 rows × 2 columns

Lemmatization and removing stopwords

Importing necessary libraries

```
#import libraries
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet
```

Creating a lemmatizing function combined with a stopword identifier

```
#lemmatizing function combined with a stopword identifier
def lemmatizer(df):
    df["length"] = df["message"].str.len()
    df["tokenized"] = df["message"].apply(word_tokenize)
    df["parts-of-speech"] = df["tokenized"].apply(nltk.tag.pos_tag)

    def str2wordnet(tag):
        conversion = {"J": wordnet.ADJ, "V": wordnet.VERB, "N": wordnet.NOUN, "R": wordnet.AD
        try:
            return conversion[tag[0].upper()]
        except KeyError:
            return wordnet.NOUN

    wnl = WordNetLemmatizer()
    df["parts-of-speech"] = df["parts-of-speech"].apply(
        lambda tokens: [(word, str2wordnet(tag)) for word, tag in tokens]
    )
    df["lemmatized"] = df["parts-of-speech"].apply(
        lambda tokens: [wnl.lemmatize(word, tag) for word, tag in tokens]
    )
    df["lemmatized"] = df["lemmatized"].apply(lambda tokens: " ".join(map(str, tokens)))

    return df
```

downloading requisite packages for the nltk library

```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

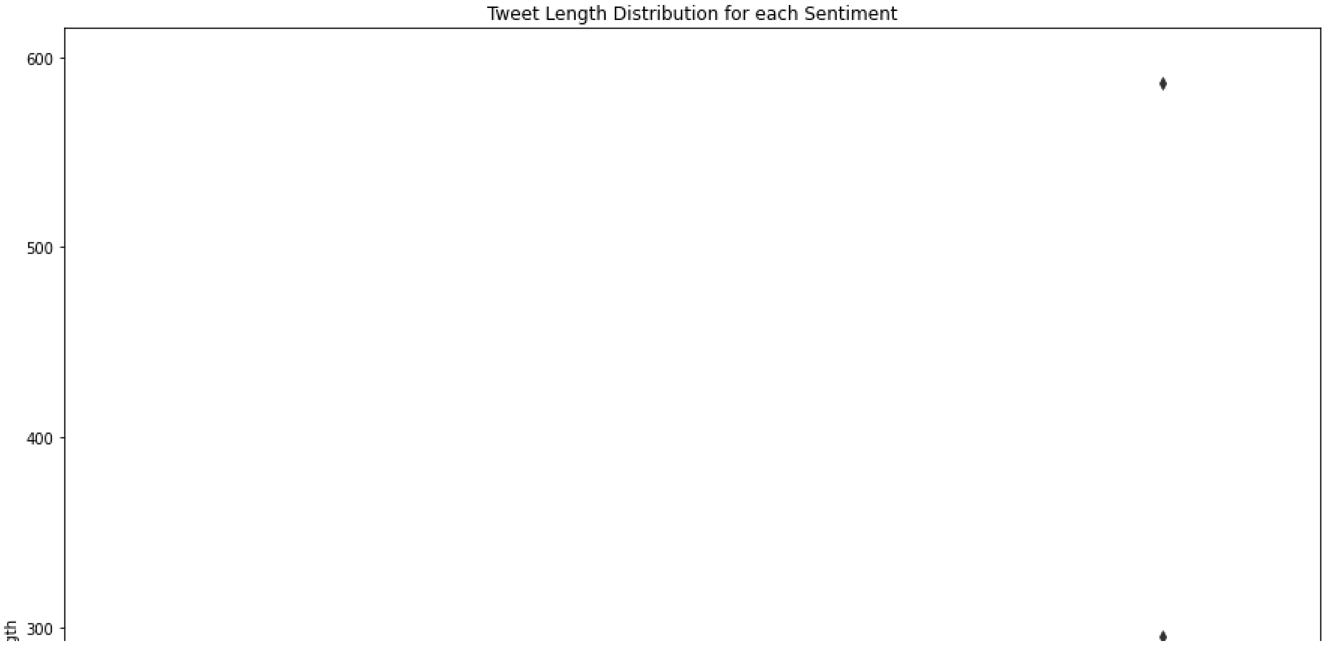
applying the lemmatizer function to our dataset and previewing the changes

```
#applying the lemmatizer function
df = lemmatizer(df)
df.head()
```

	sentiment	message	length	tokenized	parts-of-speech	lemmatized
0	Anti	climate change is an interesting hustle as it...	126	[climate, change, is, an, interesting, hustle,...	[(climate, n), (change, n), (is, v), (an, n), ...	climate change be an interesting hustle a it b...
1	Pro	watch beforetheflood right here as travels...	93	[watch, beforetheflood, right, here, as, trave...	[(watch, n), (beforetheflood, n), (right, r), ...	watch beforetheflood right here a travel the w...
		climate ...		dicaprio, s, film, an, cl...	(dicaprio, n), ...	climate c...

Visualizing the average length of tweeets after some transformation as above

```
#boxplot for visualizing the average length of tweeets
plt.figure(figsize=(15, 15))
sns.boxplot(x="sentiment", y="length", data=df, palette="Blues_d")
plt.title("Tweet Length Distribution for each Sentiment")
plt.show()
```



Observations: From the visualization above, it is evident that most tweets range from 50 to 150 characters while some are in excess of 200 characters.

Other tweets are observed to be very short, essentially less than 50 characters

Identifying the frequent words/Buzzwords

```
#import libraries
from sklearn.feature_extraction.text import CountVectorizer
```

Calculating the frequency of words for each sentiment

```
#Calculating the frequency of words for each sentiment
frequency = {}

by_sentiment = df.groupby("sentiment")
for sentiment, group in tqdm(by_sentiment):
    cv = CountVectorizer(stop_words="english")
    words = cv.fit_transform(group["lemmatized"])

    n_words = words.sum(axis=0)
    word_freq = [(word, n_words[0, idx]) for word, idx in cv.vocabulary_.items()]
    word_freq = sorted(word_freq, key=lambda x: x[1], reverse=True)

    freq = pd.DataFrame(word_freq, columns=["word", "freq"])

    frequency[sentiment] = freq.head(n=25)
```

```
to_view = pd.concat(frequency, axis=1).head(n=25)
display(to_view)
```

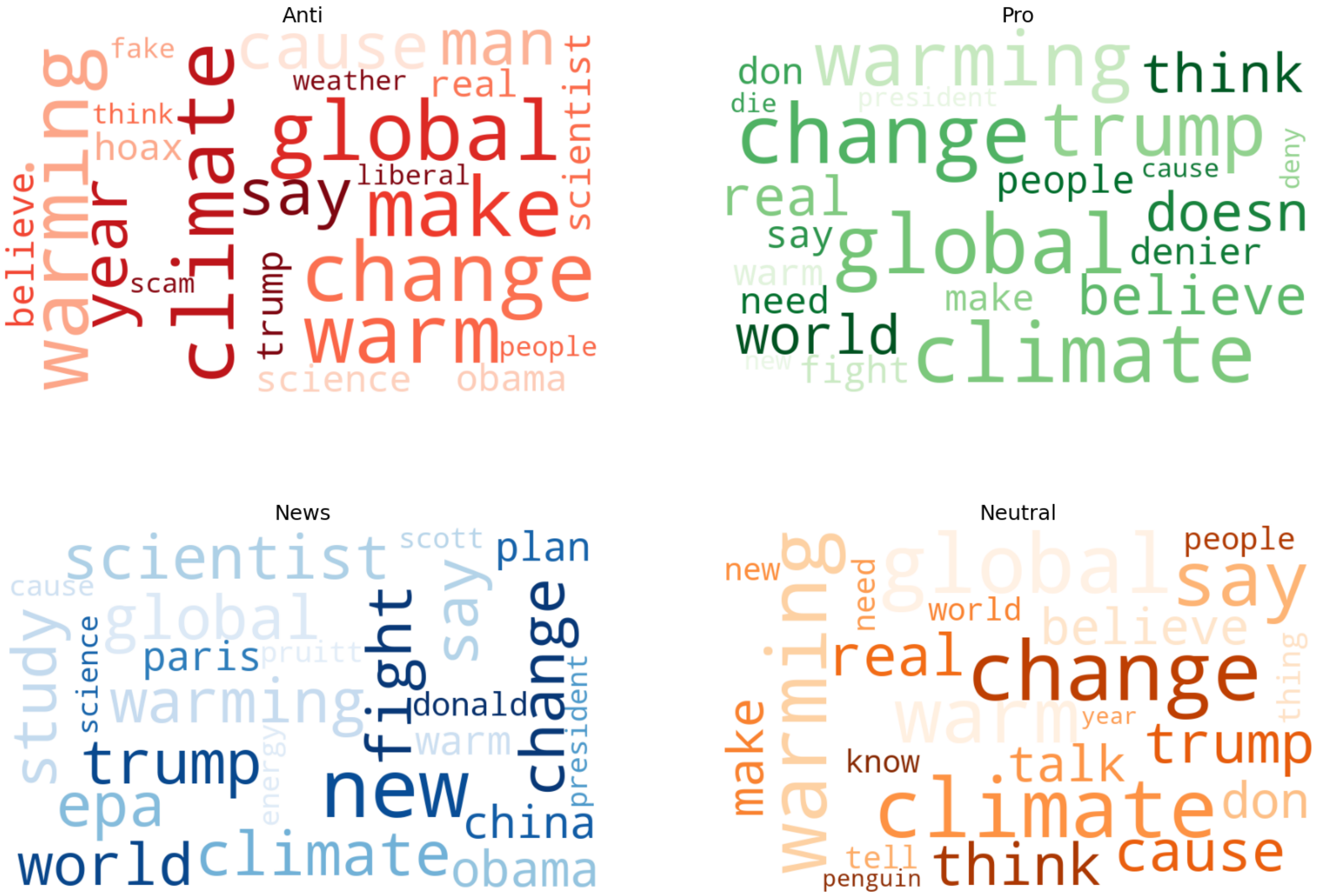
```
100%                                              4/4 [00:00<00:00, 4.56it/s]
```

	Anti		Neutral		News		Pro	
	word	**freq**	**word**	**freq**	**word**	**freq**	**word**	**freq**
0	climate	2402	climate	4652	climate	8433	climate	19827
1	change	2312	change	4621	change	8241	change	19521
2	global	1974	global	3245	trump	2086	global	4296
3	warming	1480	warming	2602	global	1418	warming	3169
4	warm	453	warm	595	warming	940	trump	2369
5	make	351	say	517	say	902	believe	2159
6	say	226	trump	492	new	577	think	1623
7	man	222	cause	331	scientist	566	doesn	1434
8	cause	220	like	330	epa	560	real	1309
9	just	208	think	327	fight	479	world	1199
10	year	202	real	295	study	416	people	1149
11	obama	196	just	291	world	377	say	1141
12	science	196	believe	285	news	367	fight	1114
13	scientist	189	make	272	obama	345	make	976
14	hoax	186	don	234	china	337	just	937
15	trump	178	talk	227	paris	315	don	930
16	believe	176	people	205	warm	311	warm	868
17	real	168	know	191	plan	288	denier	838
18	people	168	new	165	donald	281	need	834
19	liberal	161	world	163	pruitt	260	like	805
20	weather	145	tell	161	energy	242	president	781
21	fake	142	thing	156	cause	239	die	750
22	like	138	need	156	president	235	deny	733
23	think	135	penguin	155	science	233	new	732
24	scam	135	year	150	scott	227	cause	678

Building a wordcloud visualizer for each word frequency in the sentiments

```
#wordcloud visualizer for each word frequency in the sentiments
words = {sentiment: " ".join(frequency[sentiment]["word"].values) for sentiment in sentiment_
cmaps = {
    "Anti": ("Reds", 110),
    "Pro" : ("Greens", 73),
    "News": ("Blues", 0),
    "Neutral": ("Oranges", 10),
}

from wordcloud import WordCloud

wordclouds = {}
for sentiment, (cmap, rand) in tqdm(cmaps.items()):
    wordclouds[sentiment] = WordCloud(
        width=800, height=500, random_state=rand,
        max_font_size=110, background_color="white",
        colormap=cmap
    ).generate(words[sentiment])

fig, axes = plt.subplots(2, 2, figsize=(28, 20))
counter = 0

for sentiment, wordcloud in wordclouds.items():
    axes[counter // 2, counter % 2].imshow(wordcloud)
    axes[counter // 2, counter % 2].set_title(sentiment, fontsize=25)
    counter += 1

for ax in fig.axes:
    plt.sca(ax)
    plt.axis("off")

plt.show()
```

```
100%                                                    4/4 [00:00<00:00, 7.05it/s]
```



Observations:

- The top 4 buzzwords are climate, change, warming and global . This seems to indicate that a lot of the same information is being shared/viewed – this applies across all sentiments. While we can't conclude that's a result of the "filter bubble", it certainly seems like that might be a latent (hidden) cause.
- Interestingly, trump occurs across all cases. This may not be surprising given his presidency during the timeframe the Tweets were recorded – this is something that likely warrants further investigation especially along the axes of Neutral and Pro.
- Words like real, believe, think, and fight occur quite frequently in the Pro sentiment. Interestingly, both the Pro and Anti sentiment seem to be saying science and scientist, which seems indicative that both sides believe their quoting accurate, reproduced, research.
- Take a look at the table above, you'll see the http actually shows up in the Pro sentiment quite frequently. This would imply that links are being shared alongside the Tweets quite frequently. Contrast that with the other sentiments – particularly, News. Why might this be the case?

Converting the sentiment column back to number representation.

```
# sentiment column back to number representation
sentiment_name2num = {
     "Anti": 3,
     "Neutral": 0,
     "Pro" :1,
     "News": 2,
}
df["sentiment"] = df["sentiment"].apply(lambda num: sentiment_name2num[num])
df.head()
```

	sentiment	message	length	tokenized	parts-of-speech	lemmatized
0	3	climate change is an interesting hustle as it...	126	[climate, change, is, an, interesting, hustle,...	[(climate, n), (change, n), (is, v), (an, n), ...	climate change be an interesting hustle a it b...
1	1	watch beforetheflood right here as travels...	93	[watch, beforetheflood, right, here, as, trave...	[(watch, n), (beforetheflood, n), (right, r), ...	watch beforetheflood right here a travel the w...
2	1	fabulous leonardo dicaprio s film on climate ...	84	[fabulous, leonardo, dicaprio, s, film, on, cl...	[(fabulous, a), (leonardo, n), (dicaprio, n), ...	fabulous leonardo dicaprio s film on climate c...
		just watched this		[just, watched,	[(just, r),	just watch this

MODELING

```
# splitting the data into training and testing data

X= df['lemmatized'].values
y= df['sentiment'].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

```
from sklearn.feature_extraction.text import CountVectorizer

# vectorize tweets for model building
```

```
vectorizer = CountVectorizer(binary=False,stop_words='english')

# Fit the training data and then return the matrix
X_train_vec = vectorizer.fit_transform(X_train)

# Transform testing data and return the matrix. Note we are not fitting the testing data into
X_test_vec = vectorizer.transform(X_test)
```

Applying Long short-term memory (LSTM)

```
# Importing the relevant libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
#Converting the strings into integers using Tokenizer
#instantiating the tokenizer
max_vocab = 100000
tokenizer = Tokenizer(num_words=max_vocab)
tokenizer.fit_on_texts(X_train)
```

```
# checking the word index and find out the vocabulary of the dataset
wordidx = tokenizer.word_index
V = len(wordidx)
print('The size of data set vocab is: ', V)
```

```
The size of data set vocab is:  20239
```

```
# converting train and test sentences into sequences
train_sequence = tokenizer.texts_to_sequences(X_train)
test_sequence = tokenizer.texts_to_sequences(X_test)
print('Training sequence: ', train_sequence[0])
print('Testing sequence: ', test_sequence[0])
```

```
Training sequence:  [23, 1197, 5694, 4, 211, 28, 3, 17, 1005, 4, 65, 27, 23, 97, 251, 8
Testing sequence:  [23, 64, 15, 31, 10, 1, 2]
```

```
# padding the sequences to get equal length sequence
# padding the training sequence
pad_train = pad_sequences(train_sequence)
T = pad_train.shape[1]
print('The length of training sequence is: ', T)
```

```
The length of training sequence is:  33
```

```
# padding the test sequence
pad_test = pad_sequences(test_sequence, maxlen=T)
print('The length of testing sequence is: ', pad_test.shape[1])
```

```
    The length of testing sequence is:  33
```

```
#import libraries and start modeling
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D



embed_dim = 128
lstm_out = 196

model = Sequential()
model.add(Embedding(max_vocab, embed_dim,input_length = pad_train.shape[1]))
model.add(SpatialDropout1D(0.4))
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(4,activation='softmax'))
model.compile(optimizer='adam',loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logi
print(model.summary())
```

```
    Model: "sequential"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #   
    =================================================================
     embedding (Embedding)       (None, 33, 128)           12800000  
                                                                     
     spatial_dropout1d (SpatialD  (None, 33, 128)          0         
     ropout1D)                                                       
                                                                     
     lstm (LSTM)                 (None, 196)               254800    
                                                                     
     dense (Dense)               (None, 4)                 788       
                                                                     
    =================================================================
    Total params: 13,055,588
    Trainable params: 13,055,588
    Non-trainable params: 0
    _________________________________________________________________
    None
```

```
epochs = 5

history = model.fit(pad_train, y_train,
                    epochs=epochs)
```

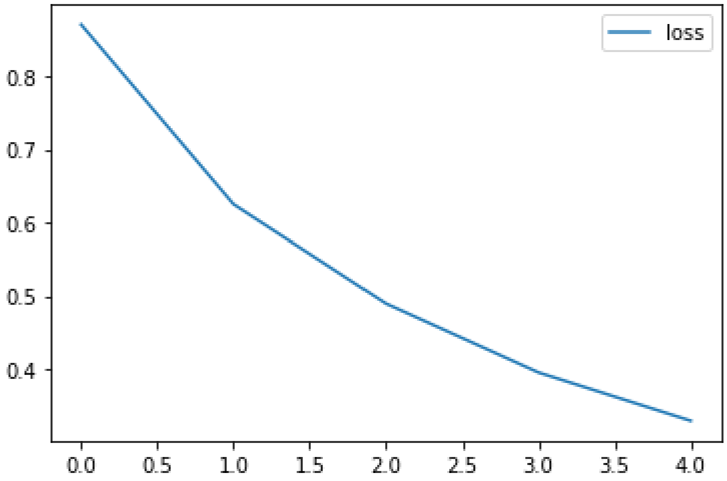
```
    Epoch 1/5
    962/962 [==============================] - 184s 188ms/step - loss: 0.8707 - accuracy: 0
    Epoch 2/5
    962/962 [==============================] - 179s 186ms/step - loss: 0.6252 - accuracy: 0
    Epoch 3/5
```

```
962/962 [==============================] - 180s 187ms/step - loss: 0.4896 - accuracy: 0
Epoch 4/5
962/962 [==============================] - 179s 186ms/step - loss: 0.3955 - accuracy: 0
Epoch 5/5
962/962 [==============================] - 179s 186ms/step - loss: 0.3294 - accuracy: 0
```

Evaluating the model.

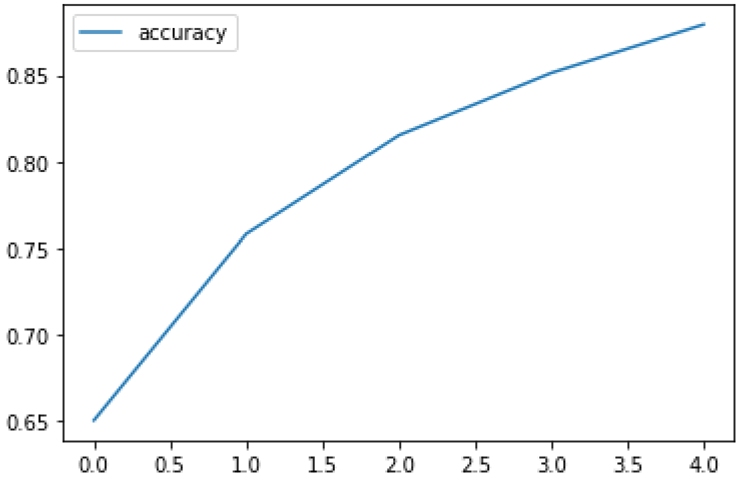
```
# plotting the loss of the model
plt.plot(history.history['loss'], label='loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fe4860f5e90>
```



```
# plotting the accuracy of the model
plt.plot(history.history['accuracy'], label= 'accuracy')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fe485faddd0>
```



The model has 88% accuracy at 5 epochs

MODEL VALIDATION

Applying Naive Bayes for model validation

```
# importing the Naive Bayes model libraries
from sklearn.naive_bayes import MultinomialNB

#Initializing and fitting the model
model = MultinomialNB().fit(X_train_vec, y_train)
```

```
#Conducting Predictions
predicted = model.predict(X_test_vec)
print(np.mean(predicted == y_test))
```

```
0.6727603732079193
```

```
#Evaluating Accuracy on Train set
print('Accuracy score: {}'.format(accuracy_score(y_test, predicted)))
```

```
Accuracy score: 0.6727603732079193
```

```
#Evaluating Accuracy on Test set
print(confusion_matrix(y_test, predicted))
```

```
[[ 750 1295  212   73]
 [ 293 5948  536   52]
 [  63  934 1756   26]
 [ 155  607   68  415]]
```

```
#Evaluating Accuracy
print(classification_report(y_test, predicted))
```

```
              precision    recall  f1-score   support

           0       0.59      0.32      0.42      2330
           1       0.68      0.87      0.76      6829
           2       0.68      0.63      0.66      2779
           3       0.73      0.33      0.46      1245

    accuracy                           0.67     13183
   macro avg       0.67      0.54      0.57     13183
weighted avg       0.67      0.67      0.65     13183
```

We can see that the accuracy score of our Naive Bayes model is 67%.

Applying Decision Trees for model validation

```
#importing the Decision Tree Classifier library

from sklearn.tree import DecisionTreeClassifier
```

```
#Initializing and fitting the model

model = DecisionTreeClassifier(max_depth=20)
model.fit(X_train_vec, y_train)
```

```
DecisionTreeClassifier(max_depth=20)
```

```
#Conducting Predictions

y_train_pred = model.predict(X_train_vec)
y_test_pred = model.predict(X_test_vec)
```

```
#Evaluating Accuracy on Train set

print(accuracy_score(y_train, y_train_pred))
confusion_matrix(y_train, y_train_pred)
```

```
0.6130689206762029
array([[ 2278,  2707,   378,    22],
       [ 1802, 13160,  1127,    44],
       [  422,  3219,  2822,    34],
       [  747,  1199,   201,   598]])
```

```
#Evaluating Accuracy on Test set

print(accuracy_score(y_test, y_test_pred))
confusion_matrix(y_test, y_test_pred)
```

```
0.5767276037320792
array([[ 872, 1239,  190,   29],
       [ 812, 5410,  575,   32],
       [ 213, 1403, 1131,   32],
       [ 352,  605,   98,  190]])
```

We can see that the accuracy score of our decision trees model is 58%.The decision trees model didn't perform better than the Naive Bayes score. Hence we'll select the Naive Bayes model in model validation.

CONCLUSION AND RECOMMENDATIONS

Conclusion

1. More people are into man-made Climate change
2. Those against man-made climate change are as a result of believes.
3. The most popular hashtags are, broadly, climate and climatechange. Which is expected, given the topic; but also, among the top 3 are relating to trump and his campaign slogan maga.
4. COP22, ParisAgreement, and Trump in the Pro sentiment are likely related to the formal process Trump's administration began to exit the Paris Agreements, where north of 200 nations pledged to reduce greenhour gas emissions, assist developing nations, and assist [poor] nations struggling with the consequences of a warming Earth.
5. Interestingly, auspol (short for Australian Politics) made the shortlist of the Pro sentiment. This is likely attributed to an assessment published quantifying the role of climate change in Australian brushfires and their increaseed risk of occuring.

Recommendation

1. Validating or training the model with other faster Tensorflow RNNs like GRU is recommended if memory is limited.