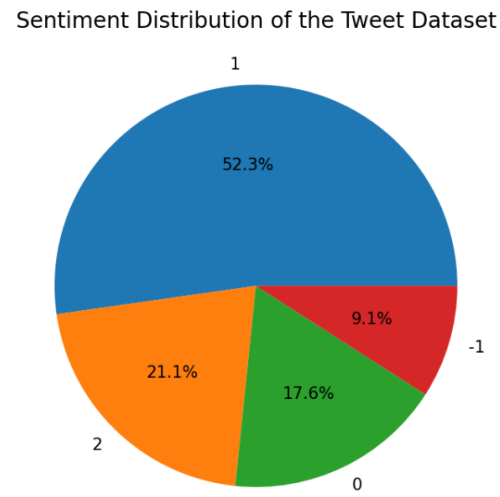
Results

▾ Distribution of Sentiments

```
plt.figure(figsize = (7, 7))
plt.pie(df.sentiment.value_counts().values, labels = df.sentiment.value_counts().index, autopct = '%2.1f%%', textprops={'fontsize': 15})
plt.title('Sentiment Distribution of the Tweet Dataset', fontsize=20)
plt.tight_layout()
plt.show()

print(' "1"  positive attitude toward the belief that the climate change is caused by the man-kinds, followed by "2"(news), "0"(neutral), and "-1"(anti).')
```



▾ Tokenization (tweets are first split into arrays of words.)

```
[10] def createTokenizedArray(sentences):

        # Initialize tokenizer and empty array to store modified sentences.
        tokenizer = RegexpTokenizer(r'\w+')
        tokenizedArray = []
        for i in range(0, len(sentences)):
            # Convert sentence to lower case.
            sentence = sentences[i].lower()

            # Split sentence into array of words with no punctuation.
            words = tokenizer.tokenize(sentence)

            # Append word array to list.
            tokenizedArray.append(words)

        # print(tokenizedArray)
        return tokenizedArray  # send modified contents back to calling function.
```

```
[12] tokenizedLi = createTokenizedArray(dfTweets)

     pd.Series(tokenizedLi[:5])
```

```
0    [tiniebeany, climate, change, is, an, interesting, hustle, as, it, was, global, warmin...
1    [rt, natgeochannel, watch, beforetheflood, right, here, as, leodicaprio, travels, the,...
2    [fabulous, leonardo, dicaprio, s, film, on, climate, change, is, brilliant, do, watch,...
3    [rt, mick_fanning, just, watched, this, amazing, documentary, by, leonardodicaprio, on...
4    [rt, cnalive, pranita, biswasi, a, lutheran, from, odisha, gives, testimony, on, effec...
dtype: object
```

Stop Word Removal

eg: "the", "a", and "an"

```
[13] # To get stop words.
     nltk.download('stopwords')

     def removeStopWords(tokenList):
         '''
         Create array of words with no punctuation or stop words.
         :param tokenList: tokenized list
         :return: array of words with no punctuation or stop words.
         '''
         stopWords = set(stopwords.words('english'))
         shorterSentences = []  # Declare empty array of sentences.

         for sentence in tokenList:
             shorterSentence = []  # Declare empty array of words in single sentence.
             for word in sentence:
                 if word not in stopWords:

                     # Remove leading and trailing spaces.
                     word = word.strip()

                     # Ignore single character words and digits.
                     if (len(word) > 1 and word.isdigit() == False):
                         # Add remaining words to list.
                         shorterSentence.append(word)
             shorterSentences.append(shorterSentence)
         return shorterSentences
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[14] tokenizedNoStopLi = removeStopWords(tokenizedLi)

     print(f"Sample sentence BEFORE removing stop words:\n{tokenizedLi[0]}")
     print(f"\n\nSample sentence AFTER removing stop words:\n{tokenizedNoStopLi[0]}")
```

```
Sample sentence BEFORE removing stop words:
['tiniebeany', 'climate', 'change', 'is', 'an', 'interesting', 'hustle', 'as', 'it', 'was', 'global', 'warming', 'but', 'the', 'planet', 'stopped', 'warming', 'for', '15', 'yes',


Sample sentence AFTER removing stop words:
['tiniebeany', 'climate', 'change', 'interesting', 'hustle', 'global', 'warming', 'planet', 'stopped', 'warming', 'yes', 'suv', 'boom']
```

Stemming

```
def stemWords(sentenceArrays):
    '''
    Removes suffixes and rebuilds the sentences.
    :param sentenceArrays: stentences list
    :return: array of sentences without suffixes
    '''
    ps = PorterStemmer()
    stemmedSentences = []
    for sentenceArray in sentenceArrays:
        stemmedArray = []  # Declare empty array of words.
        for word in sentenceArray:
            stemmedArray.append(ps.stem(word))  # Add stemmed word.

        # Convert array back to sentence of stemmed words.
        delimeter = ' '
        sentence = delimeter.join(stemmedArray)

        # Append stemmed sentence to list of sentences.
        stemmedSentences.append(sentence)
    return stemmedSentences
```

```
stemmedLi = stemWords(tokenizedNoStopLi)

print(f"Sample sentence BEFORE stemming:\n{tokenizedNoStopLi[0]}")
print(f"\nSample sentence AFTER stemming:\n{stemmedLi[0]}")
```

```
Sample sentence BEFORE stemming:
['tiniebeany', 'climate', 'change', 'interesting', 'hustle', 'global', 'warming', 'planet', 'stopped', 'warming', 'yes', 'suv', 'boom']

Sample sentence AFTER stemming:
tiniebeani climat chang interest hustl global warm planet stop warm ye suv boom
```

Vectorization

Vectorization

```
[17] def vectorizeList(stemmedList, ngramRangeStart, ngramRangeEnd):
         '''
         Creates a matrix of word vectors.
         :param stemmedList: stemmed sentence list
         :return: matrix of word vectors and vocabulary dictionary
         '''
         cv = CountVectorizer(binary=True, ngram_range=(ngramRangeStart, ngramRangeEnd))
         cv.fit(stemmedList)
         X = cv.transform(stemmedList)

         return X, cv.vocabulary_
```

```
[18] vectorizedTweets, vectorDictionary = vectorizeList(stemmedLi, 1, 1)
```

```
[19] print(f"Sample sentence #1:\n{stemmedLi[0]}")
     print(f"Sample sentence #2:\n{stemmedLi[1]}")
```

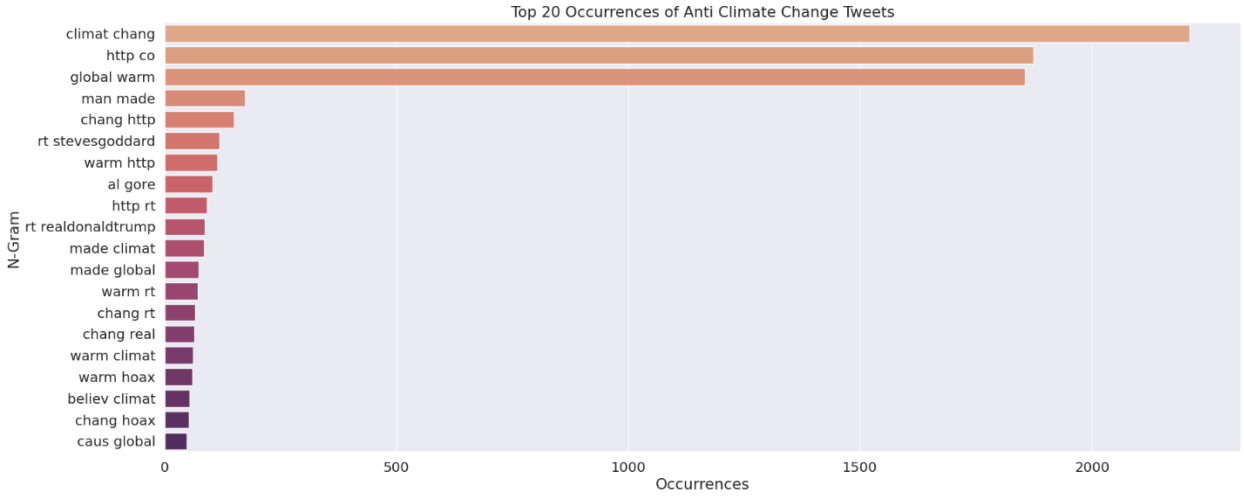
```
Sample sentence #1:
tiniebeani climat chang interest hustl global warm planet stop warm ye suv boom
Sample sentence #2:
rt natgeochannel watch beforetheflood right leodicaprio travel world tackl climat chang http co lkdehj3tnn httã
```

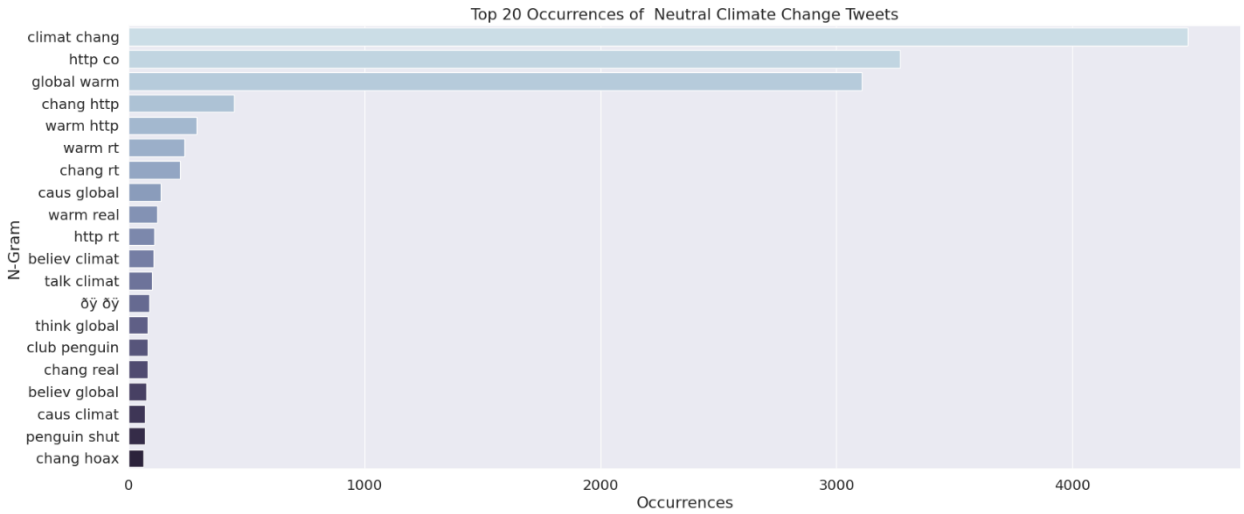
```
[20] print(f"\n#1 after vectorization:\n{vectorizedTweets[0]}")
     print(f"#2 after vectorization:\n{vectorizedTweets[1]}")
```

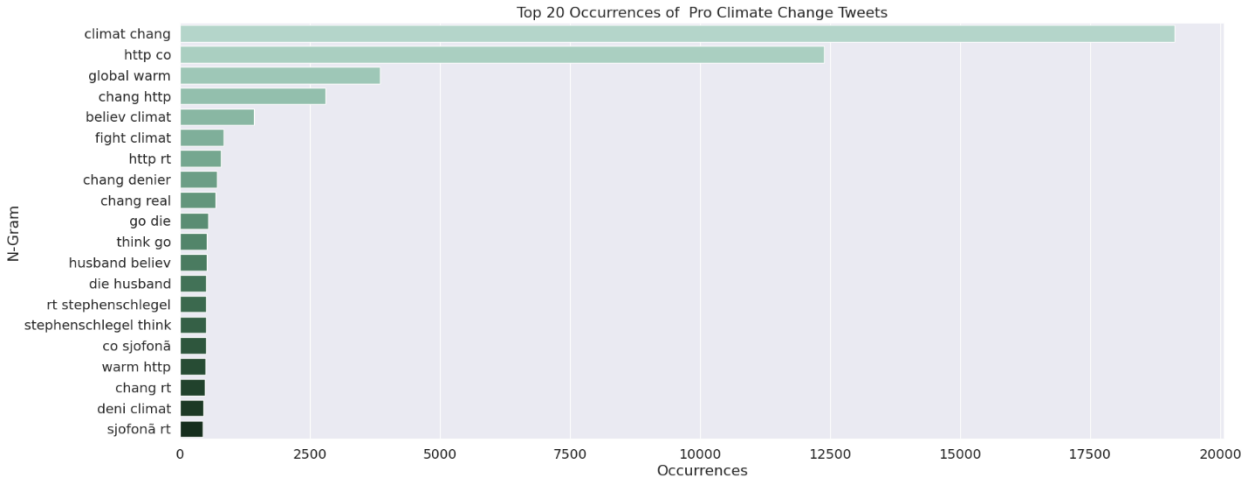
```
#1 after vectorization:
  (0, 10652)    1
  (0, 12943)    1
  (0, 13774)    1
  (0, 24524)    1
  (0, 27759)    1
  (0, 29419)    1
  (0, 45475)    1
  (0, 54598)    1
  (0, 55193)    1
  (0, 57219)    1
  (0, 61855)    1
  (0, 64839)    1
#2 after vectorization:
  (0, 9419)     1
  (0, 12943)    1
  (0, 13774)    1
  (0, 14222)    1
  (0, 27583)    1
  (0, 27591)    1
  (0, 35060)    1
  (0, 35653)    1
  (0, 40355)    1
  (0, 49581)    1
  (0, 50386)    1
  (0, 55654)    1
  (0, 57882)    1
  (0, 61927)    1
  (0, 63129)    1
```

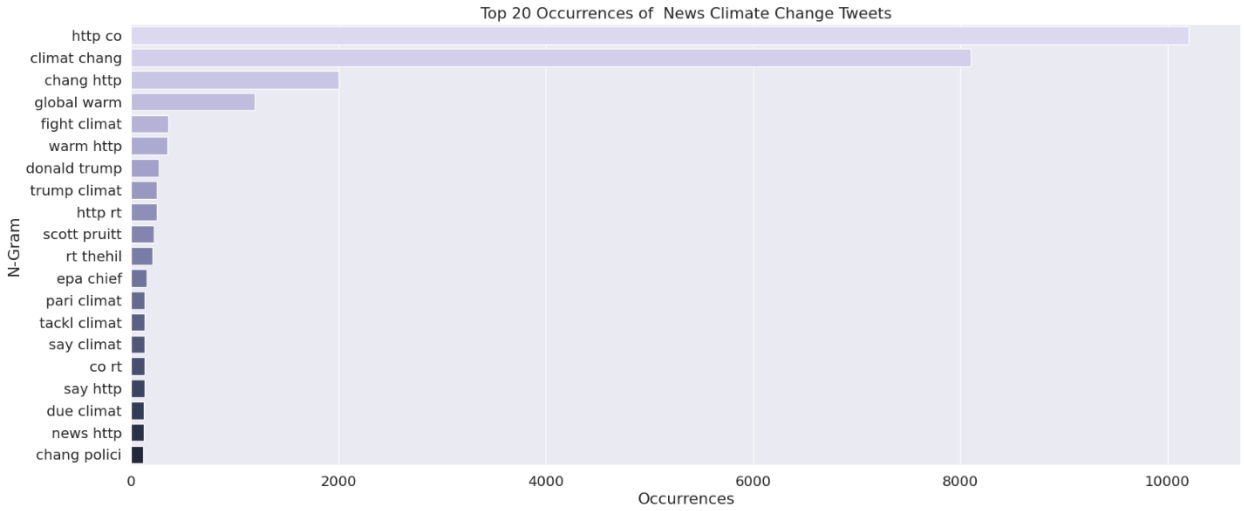
```
[21] vectorizedTweets.shape
```

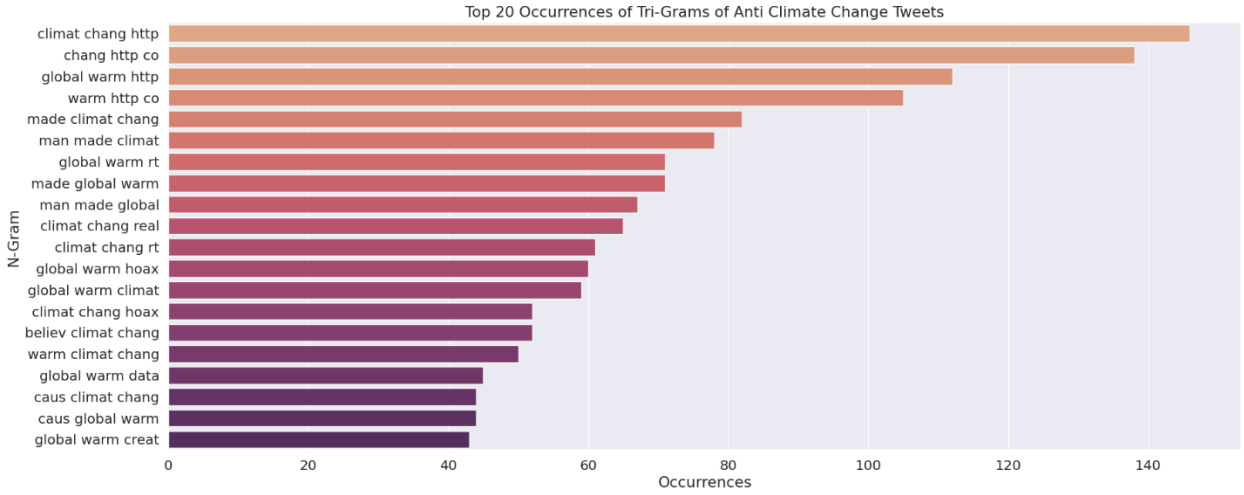
```
(43943, 67195)
```

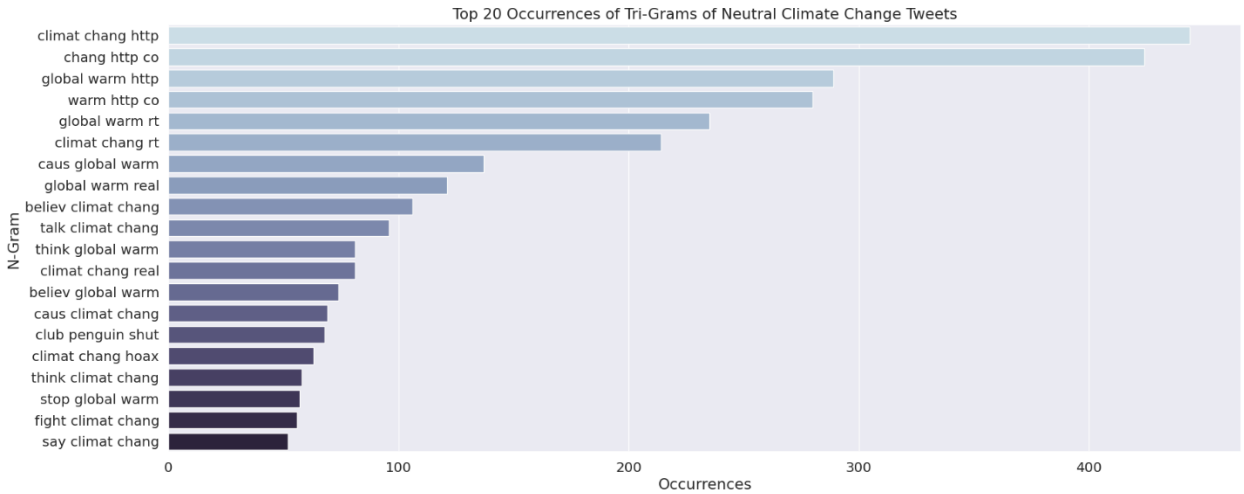

Top 20 Occurrences of Anti Climate Change Tweets
climat chang
http co
global warm
man made
chang http
rt stevesgoddard
warm http
al gore
http rt
rt realdonaldtrump
made climat
made global
warm rt
chang rt
chang real
warm climat
warm hoax
believ climat
chang hoax
caus global
N-Gram
0
500
1000
1500
2000
Occurrences

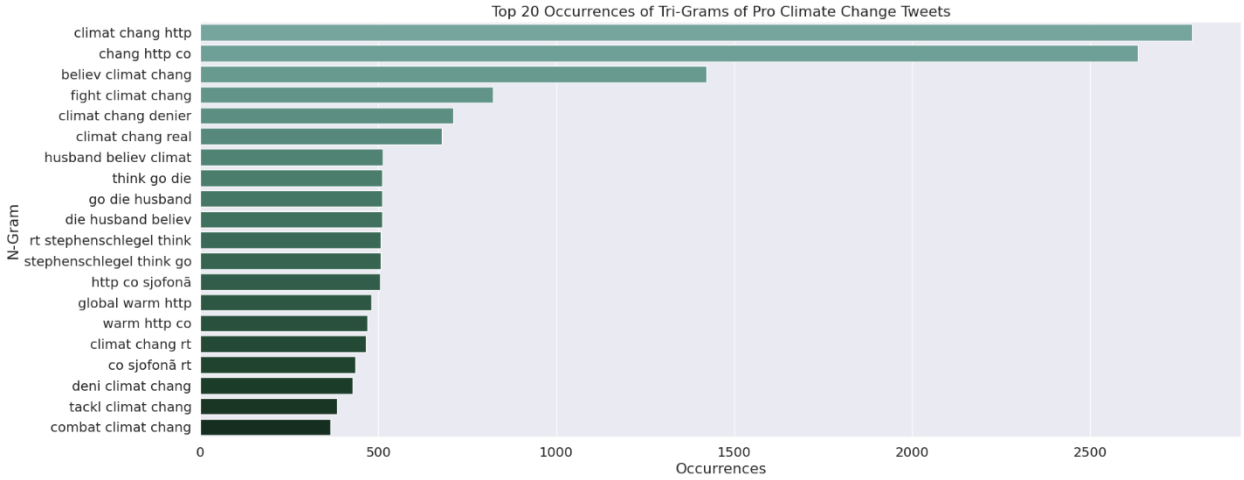

Top 20 Occurrences of Neutral Climate Change Tweets
climat chang
http co
global warm
chang http
warm http
warm rt
chang rt
caus global
warm real
http rt
believ climat
talk climat
ðÿ ðÿ
think global
club penguin
chang real
believ global
caus climat
penguin shut
chang hoax
N-Gram
0
1000
2000
3000
4000
Occurrences

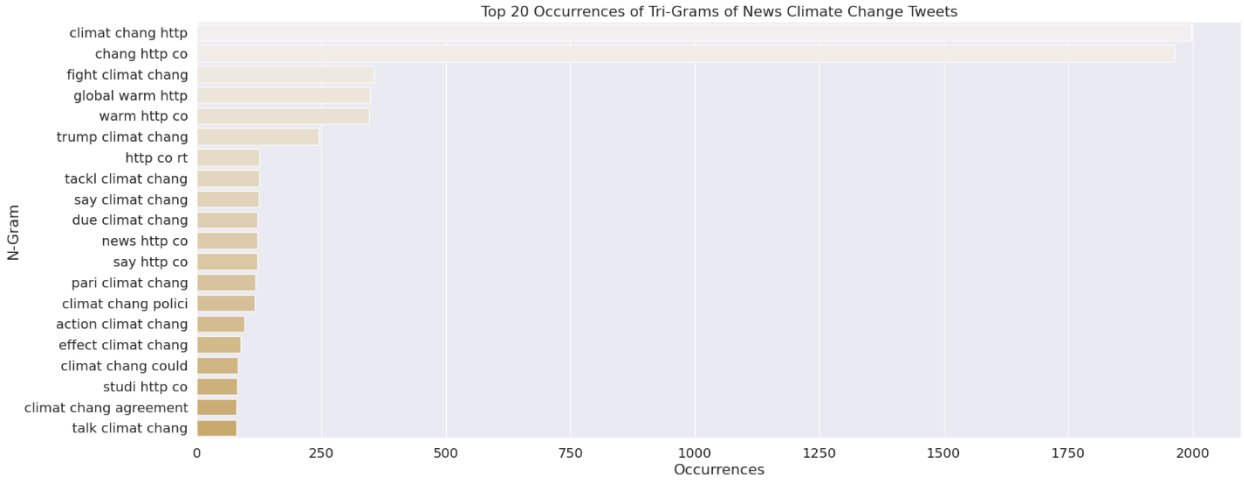

Top 20 Occurrences of Pro Climate Change Tweets
climat chang
http co
global warm
chang http
believ climat
fight climat
http rt
chang denier
chang real
go die
think go
husband believ
die husband
rt stephenschlegel
stephenschlegel think
co sjofonã
warm http
chang rt
deni climat
sjofonã rt
N-Gram
0
2500
5000
7500
10000
12500
15000
17500
20000
Occurrences


Top 20 Occurrences of News Climate Change Tweets
http co
climat chang
chang http
global warm
fight climat
warm http
donald trump
trump climat
http rt
scott pruitt
rt thehil
epa chief
pari climat
tackl climat
say climat
co rt
say http
due climat
news http
chang polici
N-Gram
0
2000
4000
6000
8000
10000
Occurrences


Top 20 Occurrences of Tri-Grams of Anti Climate Change Tweets
climat chang http
chang http co
global warm http
warm http co
made climat chang
man made climat
global warm rt
made global warm
man made global
climat chang real
climat chang rt
global warm hoax
global warm climat
climat chang hoax
believ climat chang
warm climat chang
global warm data
caus climat chang
caus global warm
global warm creat
N-Gram
0
20
40
60
80
100
120
140
Occurrences


Top 20 Occurrences of Tri-Grams of Neutral Climate Change Tweets
climat chang http
chang http co
global warm http
warm http co
global warm rt
climat chang rt
caus global warm
global warm real
believ climat chang
talk climat chang
think global warm
climat chang real
believ global warm
caus climat chang
club penguin shut
climat chang hoax
think climat chang
stop global warm
fight climat chang
say climat chang
N-Gram
0
100
200
300
400
Occurrences





Machine Learning with Logistic Regression Model

```
[39] model = LogisticRegression()
```

```
X_test, y_test, y_predicted, lrScoreDict = modelAndPredict(vectorizedTweets, df['sentiment'], model)
```

```
*** LogisticRegression ***
Accuracy:  0.7279264518478062
Precision: 0.7225203716858045
Recall:    0.7279264518478062
F1:        0.719834406019467
```

2. Tweet ML with Decision Tree

```
[45] model = DecisionTreeClassifier(max_depth=20)
```

```
[46] X_test, y_test, y_predicted, treeScoreDict = modelAndPredict(vectorizedTweets, df['sentiment'], model)
```

```
*** DecisionTreeClassifier ***
Accuracy:  0.6171490988530858
Precision: 0.6239444781518592
Recall:    0.6171490988530858
F1:        0.5677024448881892
```

3. Tweet ML with Random Forest Classifier

```
[47] model = RandomForestClassifier()
```

```
[48] X_test, y_test, y_predicted, rfScoreDict = modelAndPredict(vectorizedTweets, df['sentiment'], model)
```

```
*** RandomForestClassifier ***
Accuracy:  0.7101765883852176
Precision: 0.7144691106666974
Recall:    0.7101765883852176
F1:        0.6915729786706235
```

4. Tweet ML with K Neighbors Classifier

```
[49] model = KNeighborsClassifier()
```

```
[50] X_test, y_test, y_predicted, knnScoreDict = modelAndPredict(vectorizedTweets, df['sentiment'], model)
```

```
*** KNeighborsClassifier ***
Accuracy:  0.4344620425996723
Precision: 0.615253836784291
Recall:    0.4344620425996723
F1:        0.4473660435798679
```

5. Tweet ML with Linear Support Vector Classifier (SVC)

```
[51] model = SVC()
```

```
[52] X_test, y_test, y_predicted, svcScoreDict = modelAndPredict(vectorizedTweets, df['sentiment'], model)
```

```
*** SVC ***
Accuracy:  0.725923903149463
Precision: 0.7253793379570681
Recall:    0.725923903149463
F1:        0.7085759891802422
```

TWEET ML RESULTS of : Model Comparisons

```
[53] lrScoreDf = pd.DataFrame(lrScoreDict, index=["Logistic Regression"])
     treeScoreDf = pd.DataFrame(treeScoreDict, index=["Decision Tree"])
     rfScoreDf = pd.DataFrame(rfScoreDict, index=["Random Forest Classification"])
     knnScoreDf = pd.DataFrame(knnScoreDict, index=["K Neighbors Classification"])
     svcScoreDf = pd.DataFrame(svcScoreDict, index=["Linear Support Vector Classifier Classification"])


     clsCompDf = pd.concat([lrScoreDf, treeScoreDf, rfScoreDf,  knnScoreDf, svcScoreDf])

     clsCompDf.sort_values(by=["accuracy", "f1"], ascending = False)
```

	accuracy	recall	precision	f1
Logistic Regression	0.727926	0.727926	0.722520	0.719834
Linear Support Vector Classifier Classification	0.725924	0.725924	0.725379	0.708576
Random Forest Classification	0.710177	0.710177	0.714469	0.691573
Decision Tree	0.617149	0.617149	0.623944	0.567702
K Neighbors Classification	0.434462	0.434462	0.615254	0.447366

```
[ ]
```