1 Introduction

Disclaimer: This project is currently a proof of concept only.

1.0.1 OVERVIEW:

This notebook is where I'll be developing an algorithm for a twitterbot, HurriHelp to use. HurriHelp aims to connect folks tweeting using the #HurricaneIan hashtag with the National Disaster Distress Helpline (1-800-985-5990) and a link directing them to FEMA's page about Hurricane Ian (https://www.fema.gov/disaster/4673 (https://www.fema.gov/disaster/4673)) in real time if they are in distress. The algorithm was trained on data with sentiment analysis to judge whether or not to engage.

1.0.2 Problem: [...]

FEMA's twitter cannot respond to every tweet using the #HurricaneIan hashtag, and the people making those tweets need proactive information and resources.

1.0.3 The stakeholder: [...]

This algorithm is meant to serve folks who have been negatively impacted by Hurricane Ian and who are tweeting about it.

1.0.4 Example Use Case:

<'twitter_user'> is in distress since Hurricane Ian destroyed their home. They tweet "HurricaneIan wrecked my home of 20 years. I don't know what to do. I'm devastated." Within minutes, HurriHelp responds to their tweet saying "Hi! I'm HurriHelp, a hurricane helper bot. Here's some resources. National Disaster Distress Helpline (1-800-985-5990), for more info: https://www.fema.gov/disaster/4673" (https://www.fema.gov/disaster/4673")

1.0.5 Data Understanding:

I'll be using data that I scraped from Twitter. All these tweets contained the #HurricaneIan hashtag. The data I scraped from twitter has more features than I'm able to analyze or use currently for this project, and am saving a majority of analysis about them for future work. From the scrape, I've already filtered out Retweets and duplicates, so only the original tweets remain.

Of the features the following came directly from my twitter scrape: text, screen_name, user_description, favourite_count retweet_count, created_at, replying_to media, hashtags, urls, user_mentions, is_quote, is_retweet. Of these features I'll be using only the "text" for analysis and modeling.

The features containing sentiment analysis: text_blob, bert, vader_compound, bert_label I've engineered in a previous notebook. The first step will be making a cohesive sentiment label from text_blob, bert and vader.

The data contains 7653 rows, each representing an instance of an original tweet containing the #HurricaneIan hashtag.

Limitations:

This data is limited in that it's a small sample, compared to what's possible. I also could use a staggered sampling technique in future work. In modeling I wasn't able to get a better than 80% F1 score. It's worth investigating whether a larger amount of training data, or training data taken over a larger sample time would result in better metrics. This data is also limited in that it can't tell the difference between different negative emotions like "sad" from "mad."

Bias:

Because I used a neural network to make my labels I will *not* be using a neural network in my modeling to avoid bias.

1.1 Imports

In [1]:
```
# importing all the libraries I'll need

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

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_validate
from sklearn.metrics import plot_roc_curve, plot_confusion_matrix, confusion_matrix
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import spacy
import nltk
from nltk.collocations import *
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk import FreqDist

import string
import contractions
from cleantext import clean

from scipy.sparse import csr_matrix

from xgboost import XGBClassifier
from catboost import CatBoostClassifier

import lime
from lime import lime_text

from collections import Counter
import itertools

```

```
import pickle
```

executed in 1.79s, finished 02:10:11 2022-11-17

```
Since the GPL-licensed package `unidecode` is not installed, using Python's `unicodedata` package which yields worse results.
```

In [2]:

```
sns.set_style("white")
```

executed in 2ms, finished 02:10:11 2022-11-17

1.2 FUNCTIONS

In [3]:

```
# Making an empty list to store the classification reports my models with produce

reports = []
top_20_words = []
```

executed in 2ms, finished 02:10:11 2022-11-17

In [4]:
```
# Making a funciton to give me a ROC_AUC plot and classification report for
# each model

def evaluate(model, X_val, y_val, y_preds):
    """
    DOCSTRING:
    evaluate expects a model, a list of y_trues and associated list of
    y_predicted. it outputs a confusion matrix of PRECICSION values (noramalize is
    set to 'preds') and the associated precision, recall, f1 and support scores
    """

    lables = ['Negative Sentiment', 'Positive Sentiment']

    ROC_AUC = metrics.plot_roc_curve(estimator = model, X = X_val, y = y_val,
                                     pos_label = 0)

    report = classification_report(y_val, y_preds, target_names = lables,
                                       output_dict = False)

    dict_report = classification_report(y_val, y_preds, target_names = lables,
                                       output_dict = True)
    reports.append(dict_report)
    print(ROC_AUC);
    print('************************************************************')
    print("FULL REPORT")
    print('************************************************************')
    print(report)
```

executed in 7ms, finished 02:10:11 2022-11-17

In [5]:

```
def most_common(doc_string, sentiment_name):

    """
    DOCSTRING: intakes a string and a sentiment name and returns a plot of the
    most common words in that string, with the labels showing the sentiment name
    """
    sns.set_style('white')
    doc = nlp(doc_string)

    tokens = [token.text for token in doc]

    freqdist = FreqDist(tokens)

    most_common = freqdist.most_common(20)
    most_common_df = pd.DataFrame(most_common)
    most_common_df.head()


    plt.figure(figsize = (8, 8))
    plt.xticks(rotation = 45)

    top_20 = sns.barplot(x = 0, y = 1, data = most_common_df, color= 'blue')
    top_20.set_xlabel(f"Most Common Words in {sentiment_name}")
    top_20.set_ylabel("Frequencies")
    top_20.set_title("Top 20 Words in Hurricane Ian Tweets")
    return most_common, top_20;
```

executed in 3ms, finished 02:10:11 2022-11-17

In [6]:

```
def bigrams(sentiment_tokens, sentiment_name, name):

    """
    DOCSTRING: intakes a list of tokens and a sentiment name and returns
    a bigram plot of the most common intersection of words in those tokens,
    with the sentiment name labeled in the plot, saves plot
    """

    bigram_measures = nltk.collocations.BigramAssocMeasures()
    finder = BigramCollocationFinder.from_words(sentiment_tokens)
    scored = finder.score_ngrams(bigram_measures.raw_freq)
    top_20_bigram = pd.DataFrame(scored[:20])
    top_20_bigram.head()

    bigrams = [str(x).strip('()').title() for x in top_20_bigram[0]]
    top_20_bigram[0] = bigrams
    top_20_bigram[1] = [100 * (round(float(x), ndigits = 4)) for x in top_20_bigram[1]]

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

    top_20 = sns.barplot(x = 0, y = 1, data = top_20_bigram, color = 'green')
    top_20.set_xlabel("Bigrams")
    top_20.set_ylabel("Frequencies in Percentages")
    top_20.set_title(f"Top 20 Two-Word Combinations in {sentiment_name}")
    plt.xticks(rotation = 65)
    plt.tight_layout()
    plt.savefig(f"{name}");
```

executed in 5ms, finished 02:10:11 2022-11-17

In [7]:
```
def make_word_cloud(text, name):

    """
    DOC STRING: intakes string and makes and saves a word cloud
    """

    # Create and generate a word cloud image:
    wordcloud = WordCloud().generate(text)

    # Display the generated image:
    plt.imshow(wordcloud, interpolation='bicubic')
    plt.axis("off")
    plt.show()
    plt.savefig(f"{name}")

    wordcloud.to_file(f'{name}.png')
```
executed in 3ms, finished 02:10:11 2022-11-17

In [8]:
```
def get_cat_preds(clf, X):
    """
    DOC STRING: intakes catboost classifyer and X, and returns a list of
    predictions."""

    predict_probas = clf.predict_proba(X)
    y_preds = []
    for x in predict_probas:
        if x[0] > x[1]:
            y_preds.append(0)
        if x[0] < x[1]:
            y_preds.append(1)
    return y_preds
```
executed in 2ms, finished 02:10:11 2022-11-17

1.3 Voting

In [9]:

```python
# Loading the data

df = pd.read_csv('data_sets/ready_for_analysis.csv')
df = df.drop(columns='Unnamed: 0')
df.head()
```

executed in 107ms, finished 02:10:11 2022-11-17

Out[9]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'ir [1, 9]},
1	Republicans. can’t. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	'VoteOutEveryRepul 'ind
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexa 'indices': [18
3	Hello Everyone,\n1/3) Many Floridians face flo...	Find_and_Bind1	Amateur journalist, photographer, #bondage ent...	0	0	2022-10-03 20:18:56+00:00	NaN	False	[{'text': 'Hurrica 'indices': [112
4	Lord, please be a refuge for those in need. Gi...	shellsfaith	My name is Shelly and this is where I will be ...	1	0	2022-10-03 20:18:45+00:00	NaN	False	[{'text': 'Hurrica 'indices': [195

In [10]:
```
# Dropping the 'bert_label' as I'll be bert's numeric score instead

df = df.drop(columns = ['bert_label'])
df.head()
```

executed in 13ms, finished 02:10:11 2022-11-17

Out[10]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'ir [1, 9]},
1	Republicans. can’t. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	'VoteOutEveryRepul 'ind
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexa 'indices': [18
3	Hello Everyone,\n1/3) Many Floridians face flo...	Find_and_Bind1	Amateur journalist, photographer, #bondage ent...	0	0	2022-10-03 20:18:56+00:00	NaN	False	[{'text': 'Hurrica 'indices': [112
4	Lord, please be a refuge for those in need. Gi...	shellsfaith	My name is Shelly and this is where I will be ...	1	0	2022-10-03 20:18:45+00:00	NaN	False	[{'text': 'Hurrica 'indices': [195

In [11]:

```python
# getting the compound bert score into it's own seperate column and then removing
# the bert column

bert_scores = [x.split(' ')[3][:-2] for x in df['bert']]
df['bert_scores'] = bert_scores
df = df.drop(columns = 'bert')
df.head()
```

executed in 17ms, finished 02:10:11 2022-11-17

Out[11]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'ir [1, 9]},
1	Republicans. can’t. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	'VoteOutEveryRepul 'ind
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexa 'indices': [18
3	Hello Everyone,\n1/3) Many Floridians face flo...	Find_and_Bind1	Amateur journalist, photographer, #bondage ent...	0	0	2022-10-03 20:18:56+00:00	NaN	False	[{'text': 'Hurrica 'indices': [112
4	Lord, please be a refuge for those in need. Gi...	shellsfaith	My name is Shelly and this is where I will be ...	1	0	2022-10-03 20:18:45+00:00	NaN	False	[{'text': 'Hurrica 'indices': [195

In [12]:

```python
# making a df just of the three different scores (which are all on seperate scales)

voting_df =  [df['text_blob'], df['vader_compound'], df['bert_scores']]
voting_df = pd.DataFrame(voting_df).transpose()
voting_df
```

executed in 251ms, finished 02:10:11 2022-11-17

Out[12]:

	text_blob	vader_compound	bert_scores
0	0	-0.1531	0.352827787399292
1	0.285714	0	0.35505497455596924
2	0.625	0.9134	0.41220760345458984
3	0.5	-0.3182	0.3519584536552429
4	-0.2	0.7579	0.39440494775772095
...	...	...	...
7647	0	0.3612	0.39361315965652466
7648	0	0.8394	0.38092079758644104
7649	0.8	0.8957	0.3929575979709625
7650	0.125	-0.6476	0.3635033667087555
7651	0.13	-0.6239	0.4063631296157837

7652 rows × 3 columns

In [13]:

```
# instatiating StandardScaler and using it to transform the scoring df so all
# three scores are on the same scale

standard_scaler = StandardScaler()

scaled_voting_df = pd.DataFrame(standard_scaler.fit_transform(voting_df))
scaled_voting_df
```

executed in 12ms, finished 02:10:11 2022-11-17

Out[13]:

	0	1	2
0	-0.369372	-0.450888	-1.205824
1	0.690463	-0.162506	-1.113238
2	1.949018	1.557994	1.262663
3	1.485340	-0.761874	-1.241964
4	-1.111258	1.265091	0.522586
...	...	...	...
7647	-0.369372	0.517858	0.489670
7648	-0.369372	1.418606	-0.037966
7649	2.598168	1.524654	0.462418
7650	0.094306	-1.382339	-0.762028
7651	0.112853	-1.337697	1.019701

7652 rows × 3 columns

I'm using a SUM based voting system which doesn't care if 2 voting columns agree with each other. In the future I'd like to experiment with making something a bit more sophisticated that would take into account if two voting columns agree on a certain range and a third is way off.

In [14]:

```
# adding up the transformed scores into a final score that will be transformed
# into labels

scaled_voting_df['final_score'] = scaled_voting_df[0] + scaled_voting_df[1] + scaled_voting_df[2]
scaled_voting_df.head()
```

executed in 7ms, finished 02:10:11 2022-11-17

Out[14]:

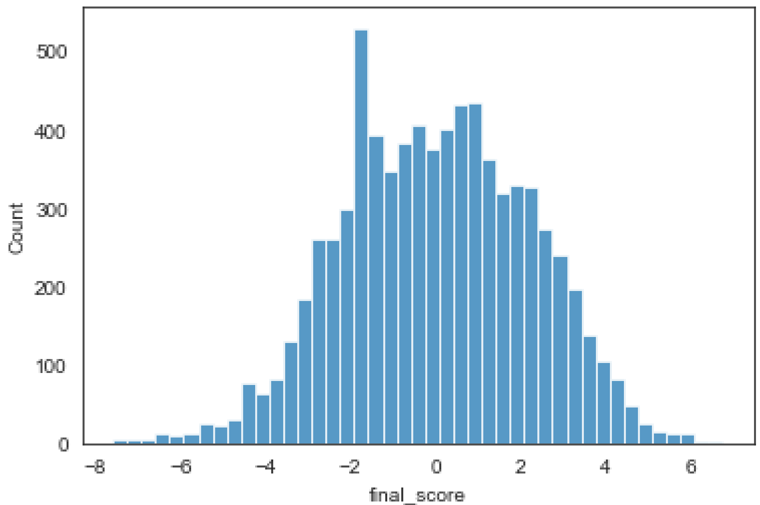
	0	1	2	final_score
0	-0.369372	-0.450888	-1.205824	-2.026085
1	0.690463	-0.162506	-1.113238	-0.585280
2	1.949018	1.557994	1.262663	4.769675
3	1.485340	-0.761874	-1.241964	-0.518498
4	-1.111258	1.265091	0.522586	0.676419

In [15]:

```
# taking a look at the distribution of final scores, we can see a regular
# distribution

sns.histplot(scaled_voting_df['final_score'])
```

executed in 202ms, finished 02:10:11 2022-11-17

Out[15]: <AxesSubplot: xlabel='final_score', ylabel='Count'>



In [16]:

```
# dropping text_blob, vader_compound and bert_score as we have our labels

df = df.drop(columns=['text_blob', 'vader_compound', 'bert_scores'])
```

executed in 5ms, finished 02:10:11 2022-11-17

In [17]:
```
df['final_score'] = scaled_voting_df['final_score']
df.head()
```
executed in 11ms, finished 02:10:11 2022-11-17

Out[17]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'ir [1, 9]},
1	Republicans. can’t. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	'VoteOutEveryRepu 'ind
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexa 'indices': [18
3	Hello Everyone,\n1/3) Many Floridians face flo...	Find_and_Bind1	Amateur journalist, photographer, #bondage ent...	0	0	2022-10-03 20:18:56+00:00	NaN	False	[{'text': 'Hurrica 'indices': [112
4	Lord, please be a refuge for those in need. Gi...	shellsfaith	My name is Shelly and this is where I will be ...	1	0	2022-10-03 20:18:45+00:00	NaN	False	[{'text': 'Hurrica 'indices': [195

2 Data Exploration

In [18]:
```
df.info()
```
executed in 10ms, finished 02:10:11 2022-11-17

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7652 entries, 0 to 7651
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   text              7652 non-null   object 
 1   screen_name       7652 non-null   object 
 2   user_description  7364 non-null   object 
 3   favourite_count   7652 non-null   int64  
 4   retweet_count     7652 non-null   int64  
 5   created_at        7652 non-null   object 
 6   replying_to       1172 non-null   object 
 7   media             7652 non-null   bool   
 8   hashtags          7652 non-null   object 
 9   urls              7652 non-null   object 
 10  user_mentions     7652 non-null   object 
 11  is_quote          7652 non-null   bool   
 12  is_retweet        7652 non-null   bool   
 13  final_score       7652 non-null   float64
dtypes: bool(3), float64(1), int64(2), object(8)
memory usage: 680.1+ KB
```

In [19]:
```
df.shape
```
executed in 3ms, finished 02:10:11 2022-11-17

Out[19]:
```
(7652, 14)
```

In [20]:
```
df['created_at'].min()
```
executed in 3ms, finished 02:10:11 2022-11-17

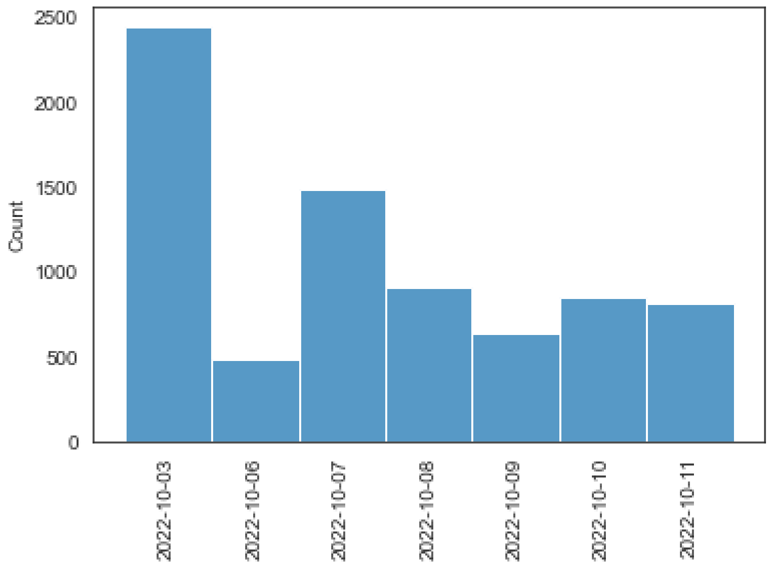
Out[20]:
```
'2022-10-03 13:49:56+00:00'
```

Started scraping at 10/03/2022 at 1:50 pm EST.

In [21]:
```
df['created_at'].max()
```
executed in 3ms, finished 02:10:11 2022-11-17

Out[21]: '2022-10-11 23:59:26+00:00'

In [22]:
```
dates = [x.split(' ')[0] for x in df['created_at']]

sns.histplot(sorted(dates), bins = 7)
plt.xticks(rotation = 90);
```
executed in 194ms, finished 02:10:11 2022-11-17



Stopped scraping at 10/11/2022 at 11:59 pm EST.

In future work, I want to really dig in and explore the correlations between sentiment and `favourite_count` or `retweet_count` and also explore all the features of this data. However, for the purposes of getting something deployed as fast as possible, with Hurricane Season on the way, I'm only going to be focusing on the text column.

2.1 Cleaning the Tweets

2.1.1 Removing Emojis

I'll be using clean_text to remove emojis from the text. After trying this process both with and without emojis and having the F1 scores come back in the same ranges, I've decided to remove the emojis, prioritizing making the text vectors less multidementional

In [23]:
```
# making a list of all tweets without emojis

no_emojis = [clean(x, no_emoji = True) for x in df['text']]
no_emojis[:10]
```

executed in 873ms, finished 02:10:12 2022-11-17

Out[23]:
```
['"#florida\'s death toll from #hurricaneian tops 100 as the search for survivors continues #fortmyers
#ftmyers #ian https://t.co/rqcyahaxtk', (https://t.co/rqcyahaxtk',)
 "republicans. can't. be. counted. on. to. do. the. right. thing.\never.\n#voteouteveryrepublican\n#vo
tethemallout\n#hurricaneian https://t.co/me3qmrztsx", (https://t.co/me3qmrztsx",)
 'leadership you can trust. make sure to like my commissioner trish becker anastasia mosquito control
 district page and share it with friends. vote by mail ballots went out today!\n#leadbyexample #vote #
politics #hurricanian #hurricaneian #hurricane #staugustine #women #mosquito https://t.co/mhs9wz1bsc',
(https://t.co/mhs9wz1bsc',)
 'hello everyone,\n1/3) many floridians face flood damage from ian without flood insurance\nhttps://t.
co/lgo1y1sfsk\n#hurricaneian #florida #flooddamage #insurance #fema #grants',
 'lord, please be a refuge for those in need. give them the comfort of your presence. bring them peace
and give them strength to carry on in the midst of trouble and loss. in your holy name, amen.\n#hurric
aneian #hurricane #prayforflorida #prayer https://t.co/m4c6nh2x1u', (https://t.co/m4c6nh2x1u',)
 "over 90% of fort myers beach, a town in south west florida, is destroyed following deadly hurricane
 ian. our crew is there talking to residents and survivors. here are some of the pictures we've captur
ed.\n#ftmyersbeach #hurricaneian #fortmyers https://t.co/hip2qrilmp", (https://t.co/hip2qrilmp",)
 'continuing to bring you updates on the aftermath of #hurricaneian\n#hurricane #ian #fl #florida #new
s #onscene #update #fortmyersbeach https://t.co/1im4sflezk', (https://t.co/1im4sflezk',)
 "@dwuhlfelderlaw @sarahburris @acosta soooo, let me get this straight. the very same ppl who say ther
e's zero chance fraud is ever committed in an election are now saying @govrondesantis is literally hid
ing hundreds of dead bodies after #hurricaneian.\nwill the real conspiracy theorists please stand up.
 gtfo!",
 'west africa really did send its best #hurricaneian https://t.co/ujwoqllapk', (https://t.co/ujwoqllap
k',)
 'you would think someone "so involved with the fishing community" would be finding out ways to help i
t #fishing #hurricaneian #florida']
```

In [24]:

```python
# making no_emojis list into into a column, I'll be doing all my text processing
# on this new column

df['no_emojis'] = no_emojis
df.head()
```

executed in 9ms, finished 02:10:12 2022-11-17

Out[24]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'ir [1, 9]},
1	Republicans. can’t. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	'VoteOutEveryRepul 'ind
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexa 'indices': [18
3	Hello Everyone,\n1/3) Many Floridians face flo...	Find_and_Bind1	Amateur journalist, photographer, #bondage ent...	0	0	2022-10-03 20:18:56+00:00	NaN	False	[{'text': 'Hurrica 'indices': [112
4	Lord, please be a refuge for those in need. Gi...	shellsfaith	My name is Shelly and this is where I will be ...	1	0	2022-10-03 20:18:45+00:00	NaN	False	[{'text': 'Hurrica 'indices': [195

2.1.2 Removing Stopwords, Numbers, URLS, Punctuation and Lemmatizing

I'll be using a combination of NLTK and spaCy along with some other smaller text libraries.

In [25]:

```python
# instantiating nlp and stopwords, adding some twitter specific and case
# specific stopwords, as well as adding all digets and punctuation to stopwords

nlp = spacy.load('en_core_web_sm')

stopwords = nlp.Defaults.stop_words

stopwords_to_add = ["\n", "\n\n", "hurricaneian", "ian", "hurricane",
                    "florida", "s", "amp", "th"]

for x in stopwords_to_add:
    stopwords.add(x)

for x in string.punctuation:
    stopwords.add(x)

for x in string.digits:
    stopwords.add(x)

print(stopwords)
```

executed in 627ms, finished 02:10:13 2022-11-17

```
{'hence', 'even', 'give', 'full', 'hereafter', 'perhaps', 'where', 'side', 'many', 'five', 'one', 'cou
ld', 'upon', 'throughout', 'using', 'nobody', 'thereafter', '~', 'yours', '5', 'are', 'nowhere', 'beca
use', 'already', '>', 'did', '’ve', 'fifty', '\n\n', 'should', 'since', 'although', 'another', 'six',
'somewhere', 'bottom', 'also', 'moreover', 'around', 'against', 'show', '|', 'fifteen', 'hurricane',
'former', '%', 'while', 'whereafter', 'please', '"', 'hurricaneian', 'eleven', 'they', '\n', 'then',
'whether', '[', 'among', 'a', 'seemed', 'otherwise', 'as', 'sometimes', 'such', 'neither', 'how', 'a
m', 'done', 'hereupon', 'rather', 'our', ']', 'beyond', 'it', 'anyhow', 'other', 'per', 'therein', 'no
ne', '0', 'whatever', 'seem', 'thereby', 'hundred', 'those', 'various', 'anyone', 'must', 'first',
'n‘t', 'amount', 'someone', 'on', 'meanwhile', 'n’t', 'why', 'there', 'i', 'becomes', 'became', 'bein
g', 'whereby', 'too', "'ll", 'and', 'few', 'take', 'will', 'me', 'ours', 'part', 'nine', 'whom', '1',
'never', 'florida', 'others', 'besides', 'you', 'behind', "'d", 'put', 'mine', 'more', 'seems', 'onc
e', 'him', 'anything', 'toward', 'doing', '&', 'her', '8', 'get', 'along', 'thus', 'though', '^', 'any
where', 'we', 'made', '_', '3', 'wherein', 'say', 'not', 'beside', 'serious', 'some', 'through', 'wer
e', 'forty', 'mostly', 'an', 'or', 'really', 'make', '’m', '\\', 'sometime', 'below', 'does', 'keep',
'any', 'within', '@', '#', 'ever', 'was', 'yourself', '{', ')', 'anyway', 'us', '2', 'somehow', '}',
'indeed', 'of', 'over', 'so', 'amp', '‘s', 'down', '‘m', 'above', 'at', 'when', 'elsewhere', 'three',
'than', 'due', 'from', '(', '7', 'else', 'used', 'enough', 'hereby', 'back', 'off', '4', 'whole', 'no
w', 'until', '/', 'latter', 'all', 'see', 'four', 'call', 'only', '+', 'herein', "'re", 'everything',
'empty', 'out', 'to', 'often', 're', 'seeming', 'well', 'eight', '‘ve', 'what', 'become', 'except',
'’re', 'no', 'cannot', 'th', 'ten', 'them', 'after', 'in', 'next', 'can', 'who', 'had', 'together', 'b
```

```
ecoming', 'latterly', 'almost', 'each', 'afterwards', '9', '’s', 'whenever', 'whereupon', 'under', 'wo
uld', 'this', 'about', 'less', 'itself', 'your', 'these', 'nothing', 'either', "n't", '<', 'just',
'!', 'own', ',', 'which', 'has', 'across', 'but', 'whereas', 'thereupon', 'its', 'between', 'somethin
g', 'however', 'least', 'two', ':', 'during', 'might', 'that', 'whence', '‘d', 'unless', 'front', 'b
e', 'herself', 'go', 'top', 'whoever', 'beforehand', 'for', 'before', 'again', 'formerly', 'been',
"'m", 'without', 'she', '-', 'everywhere', 'yourselves', 'his', 'hers', '’s', 'with', 'have', '?', 'ye
t', 's', 'name', 'both', ';', 'themselves', 'up', 'myself', "'", 'whither', 'onto', 'thence', 'twelv
e', '*', 'ourselves', 'wherever', 'every', 'nor', '’d', 'several', 'sixty', 'ca', '$', '`', '6', 'is',
'everyone', '.', 'alone', 'always', 'last', 'still', 'whose', 'if', 'further', 'much', 'namely', 'th
e', 'move', 'nevertheless', 'may', 'their', 'amongst', 'do', 'he', 'noone', '’ll', 'my', 'himself', 't
herefore', '=', 'third', 'twenty', 'into', '‘re', 'here', 'ian', 'via', 'by', 'same', 'most', 'regardi
ng', "'ve", 'towards', '‘ll', 'thru', 'very', 'quite'}
```

The following code block is a bit hairy and I'd like to come back to it to make it more organized and less nested for better performance. It does the following:

1) makes an empty list for doc tokens

2) for each row in the no_emojis column

3) make an empty list of all tokens in the tweet

4) "fix" the contractions in the tweet to keep meaning and remove punctuation ie "haven't" turns to "have not" and assign that string as "new_text"

5) make "new text" into a spaCy NLP doc

6) for each token in the spaCy NLP doc: if the token isn't in spaCy's punctuation AND if it's lemma_ (rootword) isn't a pronoun, and the token is neither a a spaCy number or a url: add token's lemma_ (rootword) to list 'tokens_in_text' after making it lower case and stripping it of surrounding spaces

7) for each token in 'tokens_in_text', filter out string.punctuation (filters more than just spaCy's punctuation)

8) for each token in 'tokens_in_text', filter out string.digits (filters more than just spaCy's 'like_numb')

9) for teach token in 'tokens_in_text', filter out all stopwords (which should include both string.punctuation and string.digits, but after running this code block a few different ways, I find added filtration with all these steps.)

10) make an object of a blank and remove all blank tokens from 'tokens_in_text'

11) adds final 'tokens_in_text' list to list of doc_tokens

12) list of doc_tokens becomes a new column in the df

In [26]:
```
contractions.add('Ft', 'Fort')
```
executed in 1ms, finished 02:10:13 2022-11-17

In [27]:

```python
# remove all punctuation, numbers, stopwords
# get lowercased lemma (rootword) from token
# make list of lemmas into new column in df

list_of_doc_tokens = []
for text in df['no_emojis']:
    tokens_in_text = []
    new_text = contractions.fix(text)
    doc = nlp(new_text)
    for token in doc:
        if not token.is_punct and token.lemma_ != '-PRON-' and not token.like_num\
        and not token.like_url:
            tokens_in_text.append(token.lemma_.lower().strip())

    tokens_in_text = [token.translate(str.maketrans('', '', string.punctuation)) for token in toke
    tokens_in_text = [token.translate(str.maketrans('', '', string.digits)) for token in tokens_in
    tokens_in_text = [token for token in tokens_in_text if token not in set(stopwords)]

    blank = ''
    tokens_in_text = [token for token in tokens_in_text if token is not blank]
    list_of_doc_tokens.append(tokens_in_text)

df['tokens'] = list_of_doc_tokens
df.head()
```

executed in 59.4s, finished 02:11:12 2022-11-17

Out[27]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha:
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'ir [1, 9]},
1	Republicans. can’t. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	'VoteOutEveryRepul 'ind
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexa 'indices': [18

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha
3	Hello Everyone,\n1/3) Many Floridians face flo...	Find_and_Bind1	Amateur journalist, photographer, #bondage ent...	0	0	2022-10-03 20:18:56+00:00	NaN	False	[{'text': 'Hurrica 'indices': [112
4	Lord, please be a refuge for those in need. Gi...	shellsfaith	My name is Shelly and this is where I will be ...	1	0	2022-10-03 20:18:45+00:00	NaN	False	[{'text': 'Hurrica 'indices': [195

2.1.3 Checking for Spammers

A bunch of tweets are bot tweets that post the same message with different links. Therefore now that the URLS are removed, I'll be checking once more for duplicates.

In [28]:

```
df = df.drop_duplicates(subset=['no_emojis', 'screen_name'])
df.shape

# this appears to have only gotten rid of 3 tweets. Instead I will explore
# how many of the rows are from the same users
```

executed in 13ms, finished 02:11:12 2022-11-17

Out[28]:

```
(7650, 16)
```

In [29]:

```
users = [x for x in df['screen_name']]

counter_ = (Counter(users))

counter_ = dict(sorted(counter_.items(), key=lambda item: item[1], reverse = True))


top_20_spammers = dict(itertools.islice(counter_.items(), 20))
print(top_20_spammers)
top_20_spammers = [key for key in top_20_spammers.keys()]
top_20_spammers
```

executed in 9ms, finished 02:11:12 2022-11-17

```
{'GaryGossipJr': 200, 'TomthunkitsMind': 108, 'foxweather': 71, 'wgcu': 49, 'AnnettemTV': 46, 'craigti
mes': 45, 'SafetyMentalst': 37, 'ReOpenChris': 32, 'ABC7Jeff': 31, 'PhilAmmann': 29, 'SRQCountyGov': 2
9, 'Fla_Pol': 28, 'HealthyCollier': 25, 'FOXCATA7': 25, 'EMS_Information': 24, 'FLSERT': 23, 'MOBILEMI
KE_': 21, 'RichFM39517086': 20, 'tampafreepress': 17, 'sdhumane': 17}
```

Out[29]:

```
['GaryGossipJr',
 'TomthunkitsMind',
 'foxweather',
 'wgcu',
 'AnnettemTV',
 'craigtimes',
 'SafetyMentalst',
 'ReOpenChris',
 'ABC7Jeff',
 'PhilAmmann',
 'SRQCountyGov',
 'Fla_Pol',
 'HealthyCollier',
 'FOXCATA7',
 'EMS_Information',
 'FLSERT',
 'MOBILEMIKE_',
 'RichFM39517086',
 'tampafreepress',
 'sdhumane']
```

In the future I'd like to sample the data by only grabbing one tweet for username or at least a way to keep the first tweets by each of the spammers but for right now I'm just going to remove those rows from these users.

BIAS: I am introducing bias into this data set here. Someone can be in distress and therefore tweeting about it frequently. Since I'll be making a lot of changes to data collection and training in the future anyway, I'm going to accept this bias as is for now and keep going.

In [30]:
```
df = df[~df['screen_name'].isin(top_20_spammers)]
df.shape
```
executed in 21ms, finished 02:11:12 2022-11-17

Out[30]: (6773, 16)

In [31]:
```
df.head()
```
executed in 10ms, finished 02:11:12 2022-11-17

Out[31]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	hasl
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'inc [1, 9]}, {
1	Republicans. can't. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	[{ 'VoteOutEveryRepubl 'indic
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexar 'indices': [180
3	Hello Everyone,\n1/3)	Find_and_Bind1	Amateur journalist,	0	0	2022-10-03	NaN	False	[{'text': 'Hurrican

In [32]:

```python
# taking a look at the most negative tweets in the data set

df['final_score'] = scaled_voting_df['final_score']
sentiment_df = df.sort_values(by = 'final_score')
for x in sentiment_df.head().values:
    print(f'TEXT: {x[0]}\n\nSCORE: {x[13]}\n \
____________________________________________________________________________\n')
```

executed in 5ms, finished 02:11:12 2022-11-17

```
TEXT: Disgusting that @MSNBC has someone like @JoyAnnReid laughing and mocking people in #Florida who died and lost everything because she hates @GovRonDeSantis   This is just horrible. MSNBC should be ashamed. #HurricaneIan

SCORE: -7.590718659081679
 ____________________________________________________________________________

TEXT: @Nextdoor is the worst for #Boomers. We went through the worst storm here #HurricaneIan. People do not have a home and they are bitching about power. #IanHitMAGA voters.

SCORE: -7.510536135686834
 ____________________________________________________________________________

TEXT: Ron DeSantis decided to play god with #immigrants' lives by flying them to #MarthasVineyard. This opened the #Karma door for #HurricaneIan to cause devastation to the people of #Florida. People died bc of his #evil actions. #KarmaIsReal
#HurricaneIanUpdate
#HurricaneIanRelief

SCORE: -7.418923571784827
 ____________________________________________________________________________

TEXT: Disgusting!! Will it ever end !! #HurricaneIan #RonDeSantis #DeSantisDestroysFlorida https://t.co/4Ev3TdyfTp (https://t.co/4Ev3TdyfTp)

SCORE: -7.367901252890476
 ____________________________________________________________________________

TEXT: Florida woman reveals devastating toll of Hurricane Ian on community
#Florida #HurricaneIan #devastating #destruction  https://t.co/JUCZhOrMIi (https://t.co/JUCZhOrMIi)

SCORE: -7.278355676013536
 ____________________________________________________________________________
```

Observations: All the most negative tweets involve political figures except the last. That's very interesting especially since we can see that they don't agree. In the future I'll want to make tweets only like the last tweet included in the target. Right now, I'm going to use the data as is as a proof of concept until I get a chance to make better labels and filter out sales tweets.

In [33]:

```python
# taking a look at the most positive tweets in the data set

for x in sentiment_df.tail().values:
    print(f'TEXT: {x[0]}\n\nSCORE: {x[13]}\n \
________________________________________________________________________________\n')
```

executed in 2ms, finished 02:11:12 2022-11-17

```
TEXT: FREE SUPPLIES for everyone in #englewood #northport #venice #portcharlotte ❗❗

TODAY from 11am-3pm

📍473 S Indiana Ave Englewood, FL

We have food, water, gas, clothes, diapers/wipes, and more!

Please share with anyone who needs relief after #HurricaneIan https://t.co/2KeWVWjG2k (https://t.co/2KeWVWjG2k)

SCORE: 6.092840465068988
 ________________________________________________________________________________

TEXT: Choose wisely, to support #HurricaneIan relief efforts https://t.co/UQJuLn8iWS (https://t.co/UQJuLn8iWS)

SCORE: 6.243772016057077
 ________________________________________________________________________________

TEXT: Download the Best #app to share your #Best #lifestyle content: https://t.co/3LwU9aPoXG (https://t.co/3LwU9aPoXG)
#fitness #yoga #tryon #tryonhaul #candy #swim #florida #sunshine #swimwear #nyfw #hurricane #hurricaneian #ytcreator  https://t.co/XIewVgosFC (https://t.co/XIewVgosFC) https://t.co/0XxQQY08k7 (https://t.co/0XxQQY08k7)

SCORE: 6.55378057826434
 ________________________________________________________________________________

TEXT: Share the #Best #lifestyle content
Download the Best #app : https://t.co/29sgyGJ4uG (https://t.co/29sgyGJ4uG)
#fitness #yoga #tryon #tryonhaul #candy #swim #florida #sunshine #swimwear #nyfw #hurricane #hurricaneian #ytcreator  https://t.co/MTgZ1DHLfa (https://t.co/MTgZ1DHLfa) https://t.co/iA9sictL7F (https://t.co/iA9sictL7F)

SCORE: 6.7302964912324565
```

```
TEXT: Share the #Best #lifestyle content
Download the Best #app : https://t.co/29sgyGJ4uG (https://t.co/29sgyGJ4uG)
#fitness #yoga #tryon #tryonhaul #candy #swim #florida #sunshine #swimwear #nyfw #hurricane #hurricane
ian #ytcreator  https://t.co/c5shVMQjz8 (https://t.co/c5shVMQjz8) https://t.co/RdttGtAwrd (https://t.c
o/RdttGtAwrd)

SCORE: 6.749120593408525
```

Observations: I'll want to go back and do more careful cleaning in the future so that the above three 'lifestyle content' posts and ones like it don't muddy the waters of the data, which appear to have gotten through my filters by posting new URLs and slightly different messages for each tweet, so they aren't recognized as duplicates. In the future, I'll need better filters to remove sales tweets. For the time being, I'm going to carry on since this won't effect the Negative Sentiment Class.

I've decided to classify any final score over 0 to be in the positive sentiment class and any final scores that were below 0 into the negative class. The negative class will be the target class.

In [34]:
```python
# making an empty list of labels, using a for loop to make my labels and making
# that list into a column
labels = []

for x in df['final_score']:
    if x > 0:
        labels.append(1)
    else:
        labels.append(0)
df['labels'] = labels
df.head()
```
executed in 14ms, finished 02:11:12 2022-11-17

Out[34]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	hasl
0	"#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'inc [1, 9]}, {
1	Republicans. can't. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	[{ 'VoteOutEveryRepubl 'indic
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexar 'indices': [180
3	Hello Everyone,\n1/3)	Find_and_Bind1	Amateur journalist,	0	0	2022-10-03	NaN	False	[{'text': 'Hurrican

In [35]:
```python
# looking at the distribution of classes, we have a pretty balenced data set

df['labels'].value_counts(normalize = True)
# Classes for the target are balenced roughly, the difference is off by about 5% which is negligab
```
executed in 4ms, finished 02:11:12 2022-11-17

Out[35]:
```
1    0.519268
0    0.480732
Name: labels, dtype: float64
```

3 EDA on Negative and Positive Sentiment

In [36]:
```
# seperating out the negative sentiment tweets and positive sentiment tweets
# into two seperate dfs

neg_df = df.loc[df['labels'] == 0]
pos_df = df.loc[df['labels'] == 1]
```
executed in 4ms, finished 02:11:12 2022-11-17

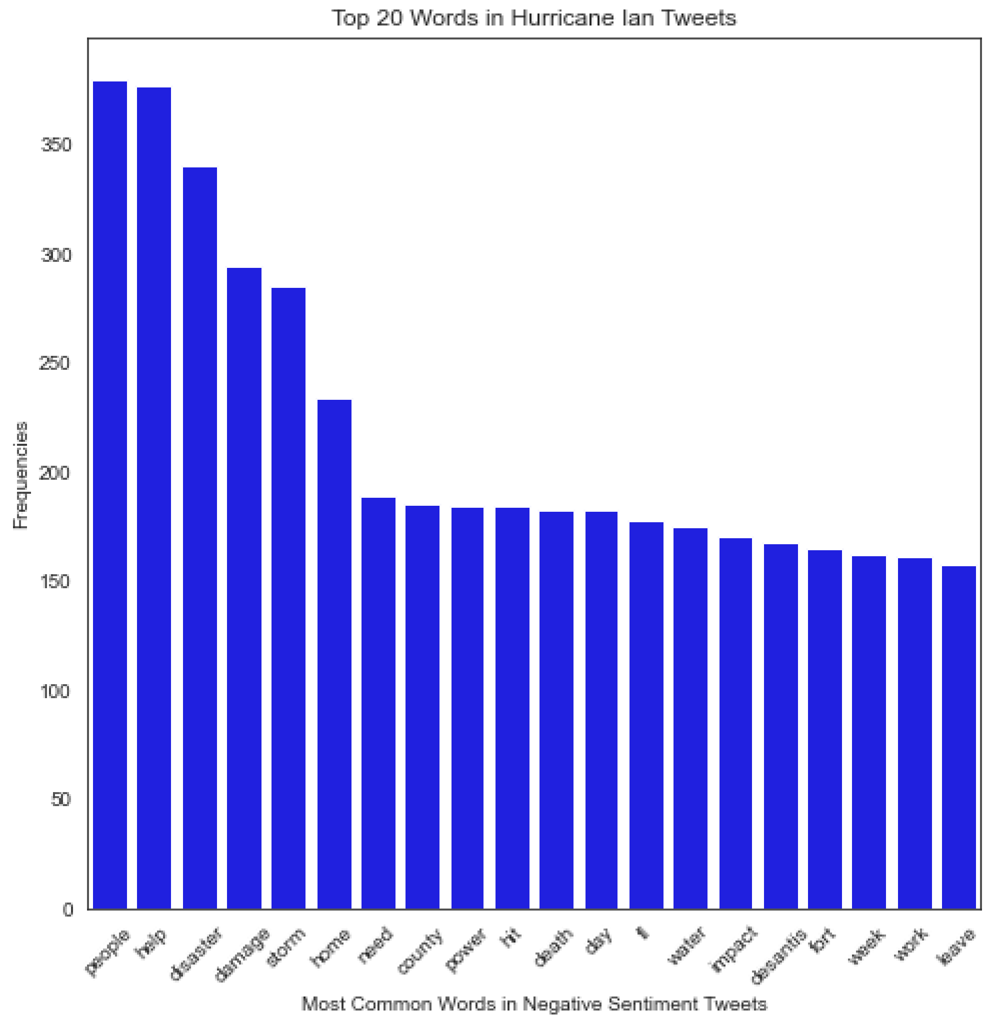
In [37]:
```
# increasing the default number of max_length so I can make sure I can proceed

nlp.max_length = 1200000
```
executed in 1ms, finished 02:11:12 2022-11-17

3.0.1 Top Words

In [38]:

```
# making a string of all the tokens in the negative df and running that string
# through the most_common function

neg_doc_string = " ".join([item for sublist in neg_df['tokens'] for item in sublist])
common_neg = most_common(neg_doc_string, "Negative Sentiment Tweets")[1]
common_neg;
```

executed in 5.76s, finished 02:11:18 2022-11-17

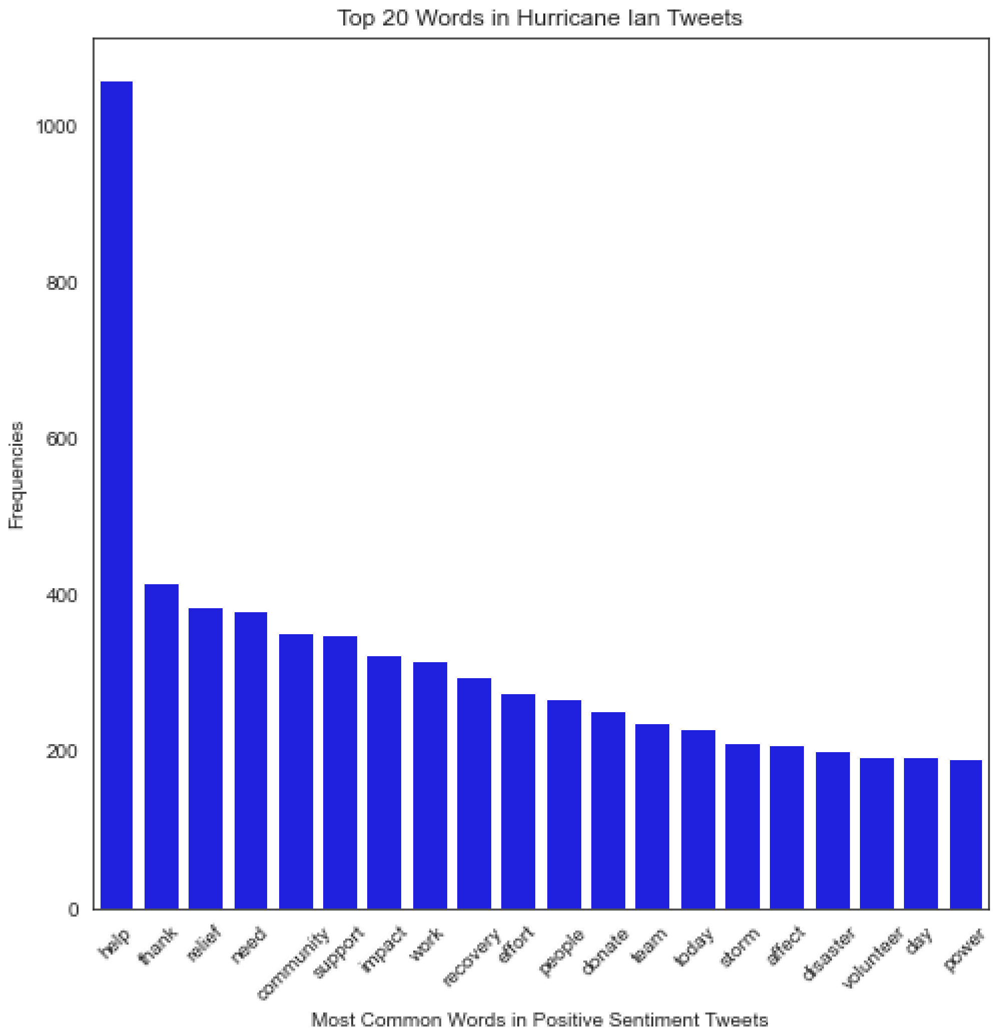


observations: 'people' and 'help' are about level with each other then a drop for 'disaster', another drop for 'damage' and 'storm' and then a drop for all the rest.

In [39]:

```
# making a string of all the tokens in the postive df and running that string
# through the most_common function

pos_doc_string = " ".join([item for sublist in pos_df['tokens'] for item in sublist])
common_posi = most_common(pos_doc_string, "Positive Sentiment Tweets")[1]
common_posi;
```

executed in 6.52s, finished 02:11:25 2022-11-17



observations: help is by far the most popular word in the positive sentiment tweets.

3.0.2 Bigrams

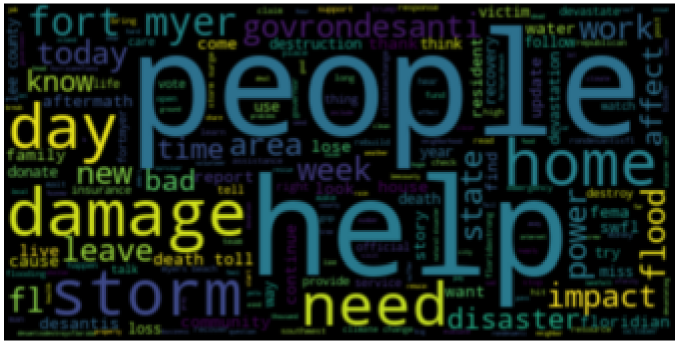
[...]

3.0.3 Word Cloud

In [42]:

```
neg_word_cloud = make_word_cloud(neg_doc_string, 'neg_word_cloud')
neg_word_cloud;
```

executed in 612ms, finished 02:11:26 2022-11-17

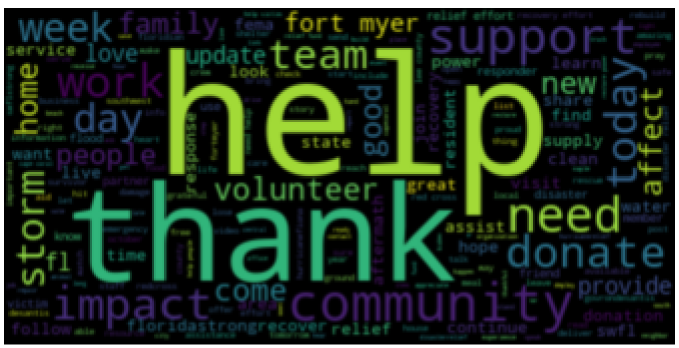


<Figure size 432x288 with 0 Axes>

In [43]:

```
posi_word_cloud = make_word_cloud(pos_doc_string, 'posi_word_cloud')
posi_word_cloud;
```

executed in 637ms, finished 02:11:27 2022-11-17



<Figure size 432x288 with 0 Axes>

4 Modeling

Method

I'll be using Random Forest, XGBoost, NaiveBayes and CatBooost to try and classify tweets into either positive or negative sentiment and then checking the labels against those predictions. For each algorithm, I'll be trying both a TFIDF vectorizer and a more simplistic CountVectorizer. I'll proceed with the algorithm and vector combination with the best F1 score relative to it's computation.

4.0.1 Train Test Split

I'll be splitting the data into 80% training data, 10% test data and 10% validation data.

In [44]:

```
df.head()
```

executed in 11ms, finished 02:11:27 2022-11-17

Out[44]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	hasl
0	“#Florida's death toll from #Hurricanelan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'inc [1, 9]}, {
1	Republicans. can’t. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	[{ 'VoteOutEveryRepubl 'indic
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexar 'indices': [180
3	Hello Everyone,\n1/3)	Find_and_Bind1	Amateur journalist,	0	0	2022-10-03	NaN	False	[{'text': 'Hurrican

In [45]:

```
token_strings = []
for x in df['tokens']:
    string = " ".join(x)
    token_strings.append(string)

df['tokens'] = token_strings
df.head()
```

executed in 13ms, finished 02:11:27 2022-11-17

Out[45]:

	text	screen_name	user_description	favourite_count	retweet_count	created_at	replying_to	media	ha
0	“#Florida's death toll from #HurricaneIan tops...	AmPowerBlog	Sports Twitter is the best Twitter. 🏈🏇🎾🛹⚾🏌 😎🚴🏐 ...	0	0	2022-10-03 20:19:43+00:00	NaN	False	[{'text': 'Florida', 'ir [1, 9]},
1	Republicans. can't. be. counted. on. to. do. ...	nivnos33	#RESISTER #Woke #Democrat #NeverGOP #VotingRig...	0	0	2022-10-03 20:19:22+00:00	NaN	False	'VoteOutEveryRepul 'ind
2	Leadership you can Trust. 🦟 Make sure to like ...	TrishTheCommish	#Commissioner, #Mom, #PublicServant, #Mosquito...	2	0	2022-10-03 20:19:09+00:00	NaN	True	[{'text': 'leadbyexa 'indices': [18
3	Hello Everyone,\n1/3) Many Floridians face flo...	Find_and_Bind1	Amateur journalist, photographer, #bondage ent...	0	0	2022-10-03 20:18:56+00:00	NaN	False	[{'text': 'Hurrica 'indices': [112
4	Lord, please be a refuge for those in need. Gi...	shellsfaith	My name is Shelly and this is where I will be ...	1	0	2022-10-03 20:18:45+00:00	NaN	False	[{'text': 'Hurrica 'indices': [195

In [46]:

```
# for the modeling purposes, I'll only be using the 'tokens' column and 'labels' column
X = df['tokens']
y = df['labels']
```

executed in 3ms, finished 02:11:27 2022-11-17

In [47]:

```
# In the first step we will split the data in training and remaining dataset
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8)

# Now since we want the valid and test size to be equal (10% each of overall data).
# we have to define valid_size=0.5 (that is 50% of remaining data)
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.5)

print(X_train.shape)
print(X_test.shape)
print(X_valid.shape)
```

executed in 4ms, finished 02:11:27 2022-11-17

```
(5418,)
(678,)
(677,)
```

In [48]:

```
# To Do: make a TFIDF function and a Count Vectorizer function
```

executed in 1ms, finished 02:11:27 2022-11-17

In [49]:

```
tfidf_vectorizer = TfidfVectorizer()

tfidf_vectorizer.fit(X_train)
X_train_tfidf_vec = tfidf_vectorizer.transform(X_train)
X_test_tfidf_vec = tfidf_vectorizer.transform(X_test)
X_valid_tfidf_vec = tfidf_vectorizer.transform(X_valid)
X_train_tfidf_vec_df.shape
```

executed in 215ms, finished 02:11:27 2022-11-17

Out[49]:

```
(5418, 12859)
```

In [50]:
```
count_vectorizer = CountVectorizer()

count_vectorizer.fit(X_train)
X_train_cv_vec = count_vectorizer.transform(X_train)
X_test_cv_vec = count_vectorizer.transform(X_test)
X_valid_cv_vec = count_vectorizer.transform(X_valid)
X_train_cv_vec_df.shape
```
executed in 293ms, finished 02:11:28 2022-11-17

Out[50]:
```
(5418, 12859)
```

In [51]:
```
print(X_train_tfidf_vec.shape)
print(X_test_tfidf_vec.shape)
print(X_valid_tfidf_vec.shape)
```
executed in 2ms, finished 02:11:28 2022-11-17

```
(5418, 12859)
(678, 12859)
(677, 12859)
```

In [52]:
```
print(X_train_cv_vec.shape)
print(X_test_cv_vec.shape)
print(X_valid_cv_vec.shape)
```
executed in 2ms, finished 02:11:28 2022-11-17

```
(5418, 12859)
(678, 12859)
(677, 12859)
```

4.0.2 Random Forest Machine

In [53]:
```
# making a GridsearchCV grid for a Random Forest Machine

rf_gscv_params = {
    'min_samples_split': [2, 3, 4, 5, 6],
    'min_samples_leaf' : [1, 2, 3, 4, 5, 6],
    'max_features' : ['sqrt'],
    'criterion': ["gini", "entropy"],
    'class_weight': ['balanced']}
```
executed in 2ms, finished 02:11:28 2022-11-17

TFIDF

In [54]:
```
# instantiating the random forest classifyer
rf_clf = RandomForestClassifier(random_state = 42)

#Applying classifyer and grid for the Gridsearch
rf_gs_tfidf = GridSearchCV(estimator = rf_clf, param_grid = rf_gscv_params,
                           scoring = 'f1', cv = 5)

#Fitting model with the training data
rf_gs_tfidf.fit(X_train_tfidf_vec, y_train)
```
executed in 14m 17s, finished 02:25:45 2022-11-17

Out[54]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

In [55]:
```
# investigating the best params
rf_gs_tfidf.best_params_
```
executed in 3ms, finished 02:25:45 2022-11-17

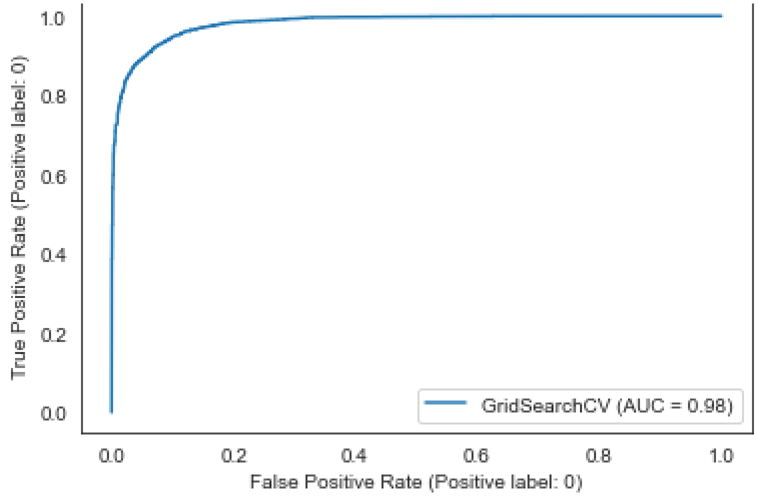
Out[55]:
```
{'class_weight': 'balanced',
 'criterion': 'entropy',
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 2}
```

In [56]:

```python
# getting my predictions from the best TFIDF random forest classyfier and evaluating
# those predictions

y_preds = rf_gs_tfidf.predict(X_train_tfidf_vec)

evaluate(rf_gs_tfidf, X_train_tfidf_vec, y_train, y_preds)
```

executed in 541ms, finished 02:25:45 2022-11-17

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)


<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd599238f70>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.94      0.90      0.92      2599
Positive Sentiment       0.91      0.94      0.93      2819

          accuracy                           0.92      5418
         macro avg       0.92      0.92      0.92      5418
      weighted avg       0.92      0.92      0.92      5418
```



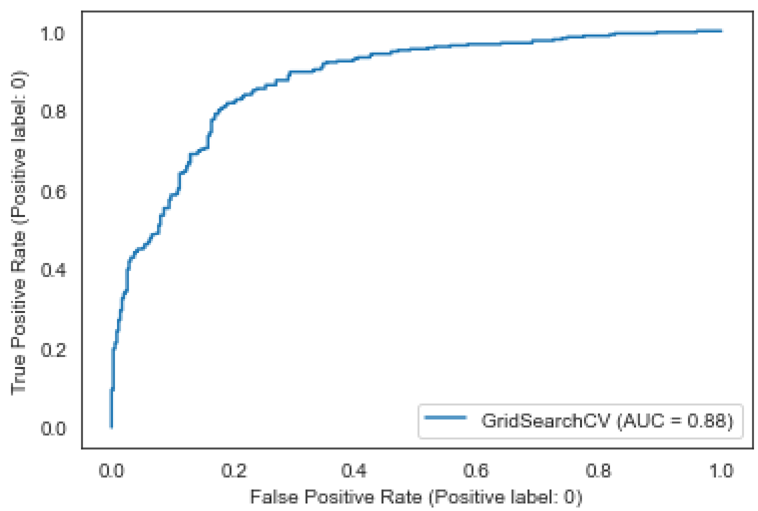
We can see this Random Forest Classyfier was able to account for 98% of the variance in the test data.

In [57]:

```
# getting the predictions of this best random forest classyfier and trying those predictions
# against the test labels for evaluation

y_preds = rf_gs_tfidf.predict(X_test_tfidf_vec)

evaluate(rf_gs_tfidf, X_test_tfidf_vec, y_test, y_preds)
```

executed in 171ms, finished 02:25:45 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd598f1e910>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.81      0.78      0.80       329
Positive Sentiment       0.80      0.83      0.82       349

          accuracy                           0.81       678
         macro avg       0.81      0.81      0.81       678
      weighted avg       0.81      0.81      0.81       678


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



We can see a 10% drop in the ability of this random forest algorithm's ability to explain the variance and our F1 score dropped by about 10% on data the model had never seen before. This is a healthy drop and not concerning.

Count Vectorizer

In [58]:
```
#Applying classyfier and grid for the Gridsearch
rf_gs_cv = GridSearchCV(estimator = rf_clf, param_grid = rf_gscv_params, scoring = 'f1', cv = 5)

#Fitting model with the training data
rf_gs_cv.fit(X_train_cv_vec, y_train)
```
executed in 12m 45s, finished 02:38:31 2022-11-17

Out[58]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [59]:
```
# investigating best params

rf_gs_cv.best_params_
```
executed in 2ms, finished 02:38:31 2022-11-17

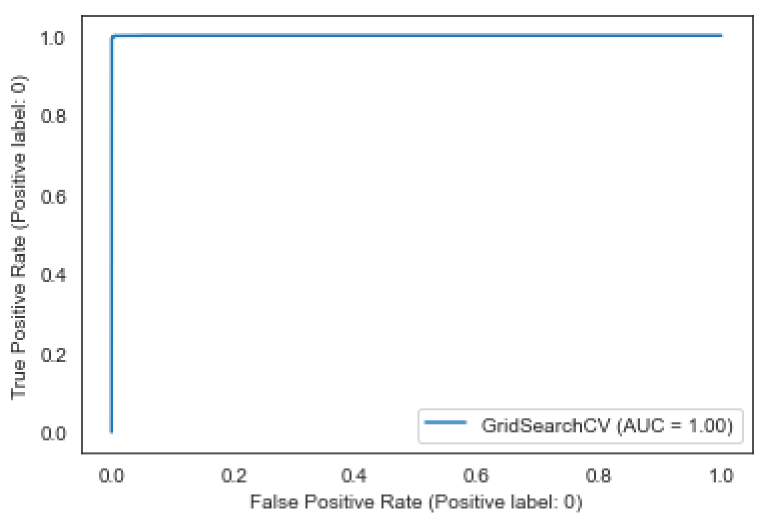
Out[59]:
```
{'class_weight': 'balanced',
 'criterion': 'entropy',
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 4}
```

In [60]:

```python
# getting predictions to try against the training labels for evaluation

y_preds = rf_gs_cv.predict(X_train_cv_vec)

evaluate(rf_gs_cv, X_train_cv_vec, y_train, y_preds)
```

executed in 589ms, finished 02:38:31 2022-11-17

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)

<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd59937b880>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       1.00      1.00      1.00      2599
Positive Sentiment       1.00      1.00      1.00      2819

          accuracy                           1.00      5418
         macro avg       1.00      1.00      1.00      5418
      weighted avg       1.00      1.00      1.00      5418
```



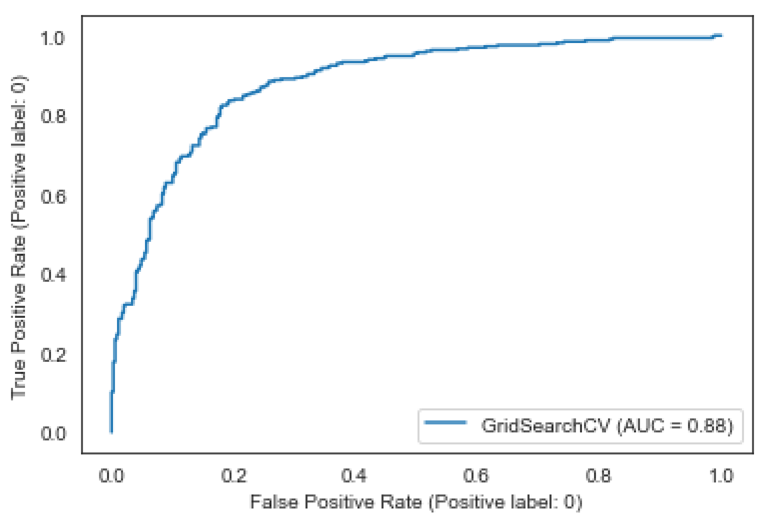
We can see this algorithm can explain 100% of the variance in the training data.

In [61]:

```python
# Getting predictions for this random forest on test data for evaluation

y_preds = rf_gs_cv.predict(X_test_cv_vec)

evaluate(rf_gs_cv, X_test_cv_vec, y_test, y_preds)
```

executed in 192ms, finished 02:38:32 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd598a32730>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.81      0.80      0.81       329
Positive Sentiment       0.81      0.83      0.82       349

          accuracy                           0.81       678
         macro avg       0.81      0.81      0.81       678
      weighted avg       0.81      0.81      0.81       678


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



This model also had a 10% drop in it's ability to explain the variance in the data, and our F1 score had another 10% healthy drop.

4.0.3 XG BOOST Machine

In [62]:
```
# Importing Counter to do calculations for the weights for the model,
# while this data is relatively balenced, this can't hurt

counter = Counter(y)
# estimate scale_pos_weight value
estimate = counter[0] / counter[1]
print('Estimate: %.3f' % estimate)
```
executed in 3ms, finished 02:38:32 2022-11-17

```
Estimate: 0.926
```

In [63]:
```
#Setting of parameters for gridsearch to use
xgb_grid = {
    'learning_rate': [0.1, 0.3, 0.5, 0.7],
    'max_depth': [10, 11, 12, 13, 14],
    'min_child_weight': [1, 2],
    'subsample': [0.3, 0.5, 0.7],
    'n_estimators': [20, 25, 30]
}
```
executed in 2ms, finished 02:38:32 2022-11-17

TFIDF

In [64]:
```
# instatiating the xgb classifyer
xg_clf = XGBClassifier()

#Applying classifyer and grid for the Gridsearch
xg_gs_tfidf = GridSearchCV(estimator = xg_clf, param_grid = xgb_grid, scoring = 'f1', cv = 5)

#Fitting model with the training data
xg_gs_tfidf.fit(X_train_tfidf_vec, y_train)
```
executed in 7m 48s, finished 02:46:20 2022-11-17

Out[64]:
```
▸ GridSearchCV
▸ estimator: XGBClassifier
  ▸ XGBClassifier
```

In [65]:
```
# investigating the best params

xg_gs_tfidf.best_params_
```
executed in 3ms, finished 02:46:20 2022-11-17

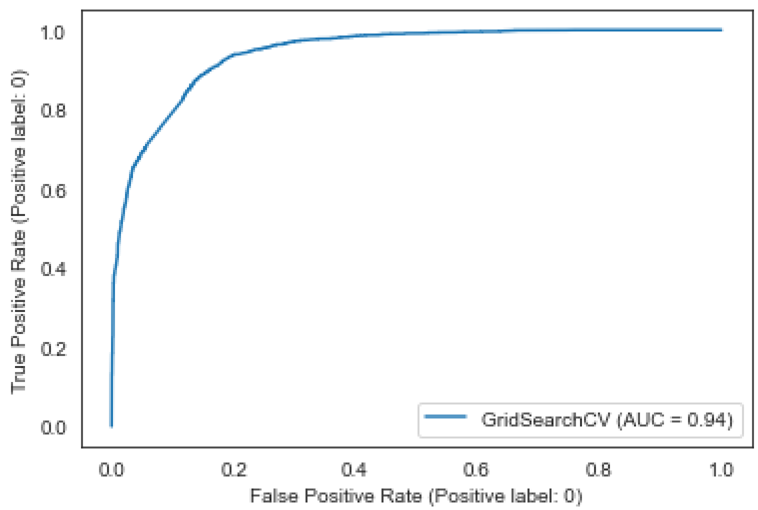
Out[65]:
```
{'learning_rate': 0.5,
 'max_depth': 11,
 'min_child_weight': 1,
 'n_estimators': 25,
 'subsample': 0.7}
```

In [66]:

```
# getting predictions to try against the training labels

y_preds = xg_gs_tfidf.predict(X_train_tfidf_vec)
evaluate(xg_gs_tfidf, X_train_tfidf_vec, y_train, y_preds)
```

executed in 224ms, finished 02:46:20 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd59990e580>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.84      0.90      0.87      2599
Positive Sentiment       0.90      0.84      0.87      2819

          accuracy                           0.87      5418
         macro avg       0.87      0.87      0.87      5418
      weighted avg       0.87      0.87      0.87      5418


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



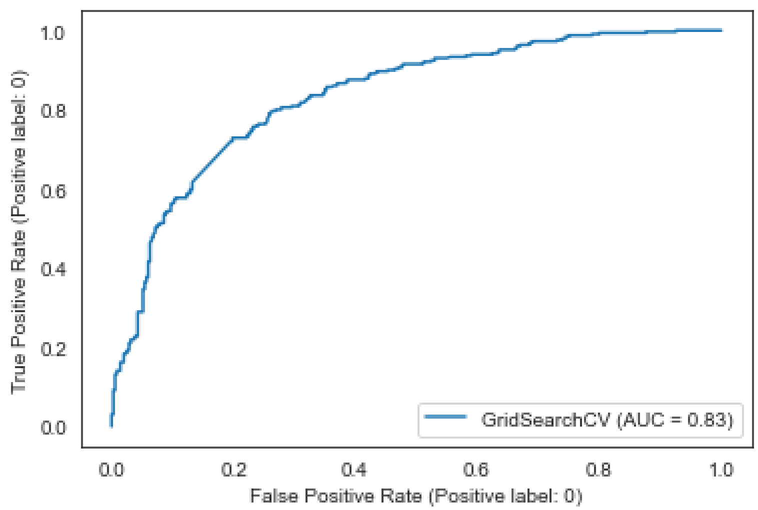
We can see the XGBoost algothim on TFIDF data was able to explain about 96% of the variance, but the F1 score is down from that AUC score by about 7%.

In [67]:

```
# getting predictions for XBGoost classyfier on TFIDF test data to try against
# the labels for evaluation

y_preds = xg_gs_tfidf.predict(X_test_tfidf_vec)
evaluate(xg_gs_tfidf, X_test_tfidf_vec, y_test, y_preds)
```

executed in 140ms, finished 02:46:20 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd5989c1a90>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.74      0.76      0.75       329
Positive Sentiment       0.77      0.75      0.76       349

          accuracy                           0.76       678
         macro avg       0.76      0.76      0.76       678
      weighted avg       0.76      0.76      0.76       678


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



The XGBoost model on the TFIDF training data could explain about 85% of the variance, down 10% from the training data. Likewise, this model's F1 score is down from 89% to about 78% in a healthy drop.

Count Vectorizer

In [68]:
```
#Applying classyfier and grid for the Gridsearch
xg_gs_cv = GridSearchCV(estimator = xg_clf, param_grid = xgb_grid, scoring = 'f1', cv = 5)

#Fitting model with the training data
xg_gs_cv.fit(X_train_cv_vec, y_train)
```
executed in 39m 30s, finished 03:25:50 2022-11-17

Out[68]:
```
▸ GridSearchCV
▸ estimator: XGBClassifier
  ▸ XGBClassifier
```

In [69]:
```
# investigating the best params
xg_gs_cv.best_params_
```
executed in 3ms, finished 03:25:50 2022-11-17

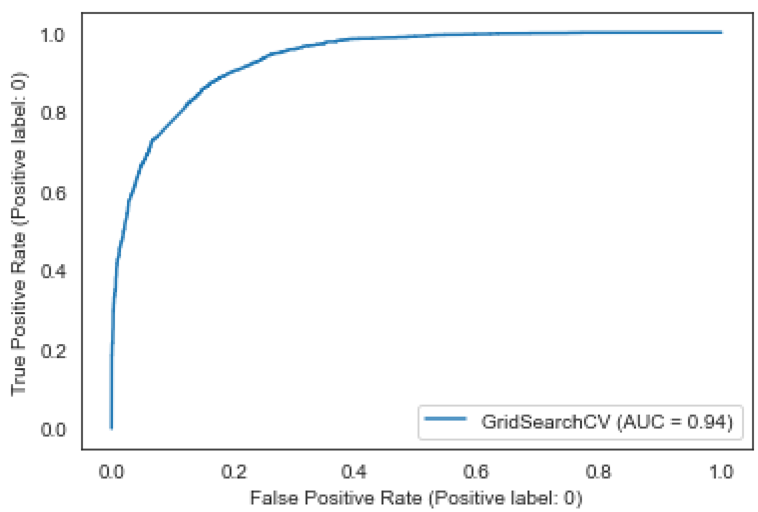
Out[69]:
```
{'learning_rate': 0.7,
 'max_depth': 12,
 'min_child_weight': 1,
 'n_estimators': 25,
 'subsample': 0.7}
```

In [70]:

```
# getting y predictions for classyfier to try against training lables for evaluation

y_preds = xg_gs_cv.predict(X_train_cv_vec)
evaluate(xg_gs_cv, X_train_cv_vec, y_train, y_preds)
```

executed in 197ms, finished 03:25:50 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd5eb13ea90>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.83      0.87      0.85      2599
Positive Sentiment       0.88      0.84      0.86      2819

          accuracy                           0.85      5418
         macro avg       0.85      0.85      0.85      5418
      weighted avg       0.85      0.85      0.85      5418



/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



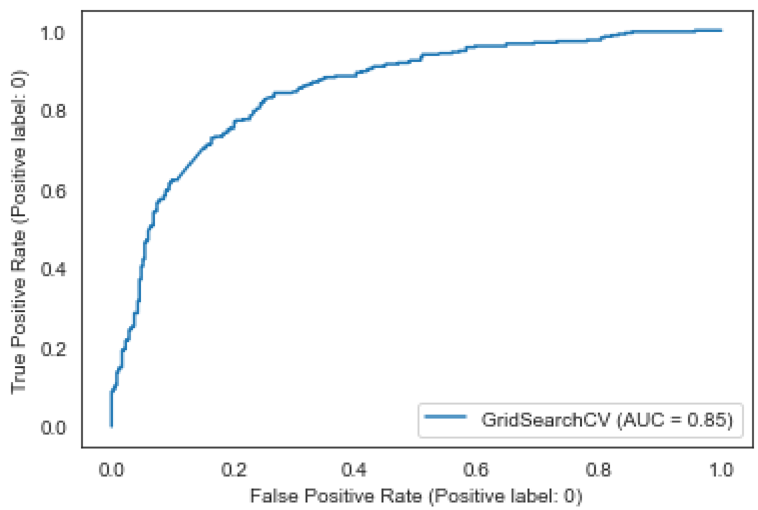
The XGBoost model can expalin about the same amount of Variance in the training data when working on Count Vectorized data, likewise the F1 score is about the same, both only loosing about 1%.

In [71]:

```
# getting predictions for this classyfier on test data to try aginst labels for
# Evaluation

y_preds = xg_gs_cv.predict(X_test_cv_vec)
evaluate(xg_gs_cv, X_test_cv_vec, y_test, y_preds)
```

executed in 178ms, finished 03:25:50 2022-11-17

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)

<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd5eb13ee80>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.78      0.76      0.77       329
Positive Sentiment       0.78      0.80      0.79       349

          accuracy                           0.78       678
         macro avg       0.78      0.78      0.78       678
      weighted avg       0.78      0.78      0.78       678
```



We can see the same healthy drop of about 10% in the AUC and F1 score for this model as well.

4.0.4 Naive Bayes Machine

TFIDF

In [72]:
```
# for the multinomail naive bayes algorithm I will not be trying GridSearch,
# but instead be using the out-of-the-box Multinomial Naive Bayes algothim

nb_clf = MultinomialNB()

#Fitting model with the training data
nb_clf.fit(X_train_tfidf_vec.toarray(), y_train)
```

executed in 343ms, finished 03:25:51 2022-11-17

Out[72]:
```
▼ MultinomialNB
MultinomialNB()
```

Since I didn't gridsearch this alorthim, training the algorithm is extreamly quick compared to the other algorithms.

In [73]:
```
# investigating the default params

nb_clf.get_params()
```

executed in 2ms, finished 03:25:51 2022-11-17

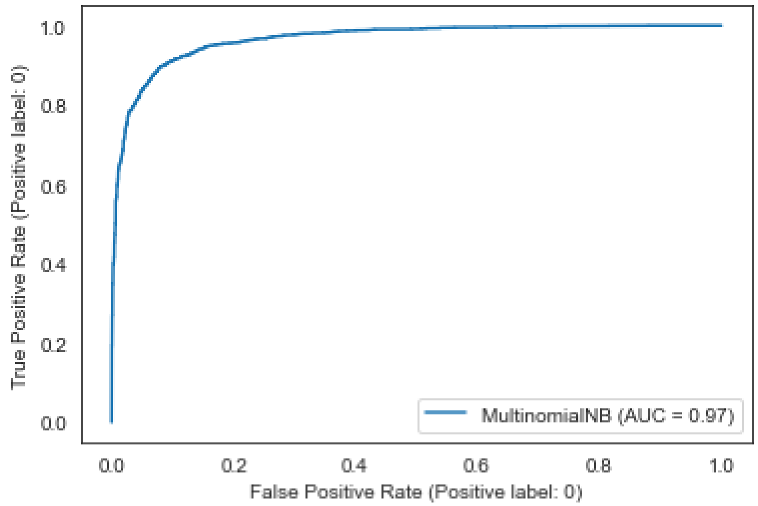
Out[73]:
```
{'alpha': 1.0, 'class_prior': None, 'fit_prior': True}
```

In [74]:

```python
# getting predictions for the Naive Bayes machine to try against the TFIDF training data
# for evaluation

# the Naive Bayes algorthim expects an array, not a sparse DF, hense the .toarray()

y_preds = nb_clf.predict(X_train_tfidf_vec.toarray())
evaluate(nb_clf, X_train_tfidf_vec.toarray(), y_train, y_preds)
```

executed in 603ms, finished 03:25:51 2022-11-17

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)

<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd59921a820>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.93      0.86      0.89      2599
Positive Sentiment       0.88      0.94      0.91      2819

          accuracy                           0.90      5418
         macro avg       0.90      0.90      0.90      5418
      weighted avg       0.90      0.90      0.90      5418
```



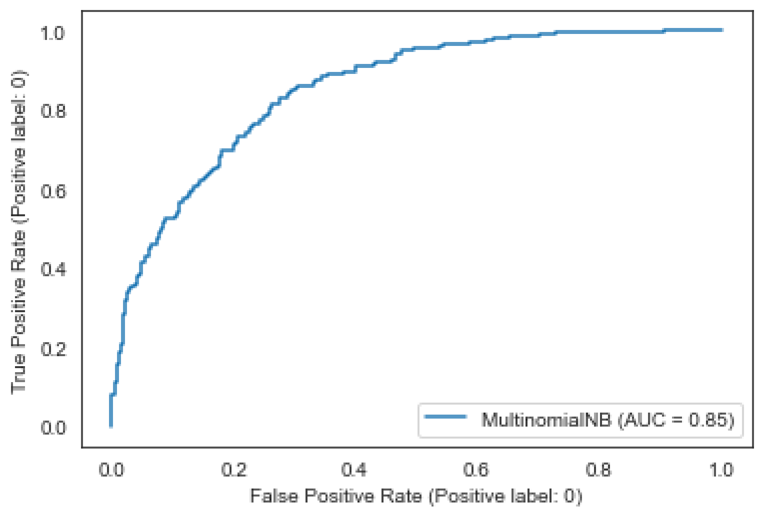
Here, the Naive Bayes algorithm was able to account for about 96% of the variance with only a 5% drop from there in F1 score.

In [75]:

```python
# getting the predictions the classyfier made to try against the test labels
# for evaluation

y_preds = nb_clf.predict(X_test_tfidf_vec.toarray())
evaluate(nb_clf, X_test_tfidf_vec.toarray(), y_test, y_preds)
```

executed in 158ms, finished 03:25:51 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd59921a550>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.78      0.66      0.72       329
Positive Sentiment       0.72      0.83      0.77       349

          accuracy                           0.75       678
         macro avg       0.75      0.74      0.74       678
      weighted avg       0.75      0.75      0.74       678


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



In the test data, the algorthim was able to explain 88% of the variance and the F1 score is about 80%, which is similar to other, much harder to train algorithms. Naive Bayes' positive advantages aren't in the F1 score or AUC score but in it's lightweight nature and quickness.

Count Vectorizer

In [76]:
```
#Fitting model with the training data
nb_clf.fit(X_train_cv_vec.toarray(), y_train)
```
executed in 579ms, finished 03:25:52 2022-11-17

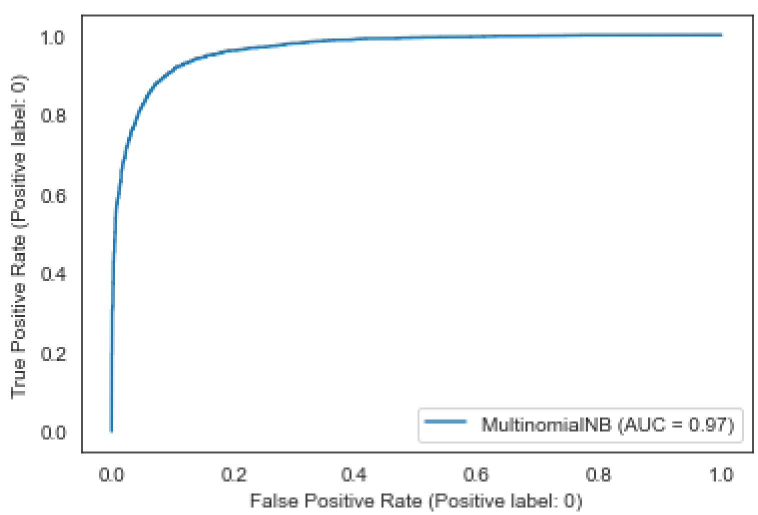
Out[76]:
```
▼ MultinomialNB
MultinomialNB()
```

In [77]:

```
# Getting predictions of algorithm on training data to test against labels for evaluation

y_preds = nb_clf.predict(X_train_cv_vec.toarray())
evaluate(nb_clf, X_train_cv_vec.toarray(), y_train, y_preds)
```

executed in 1.01s, finished 03:25:53 2022-11-17

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)

<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd598a32970>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.91      0.88      0.90      2599
Positive Sentiment       0.89      0.92      0.91      2819

          accuracy                           0.90      5418
         macro avg       0.90      0.90      0.90      5418
      weighted avg       0.90      0.90      0.90      5418
```

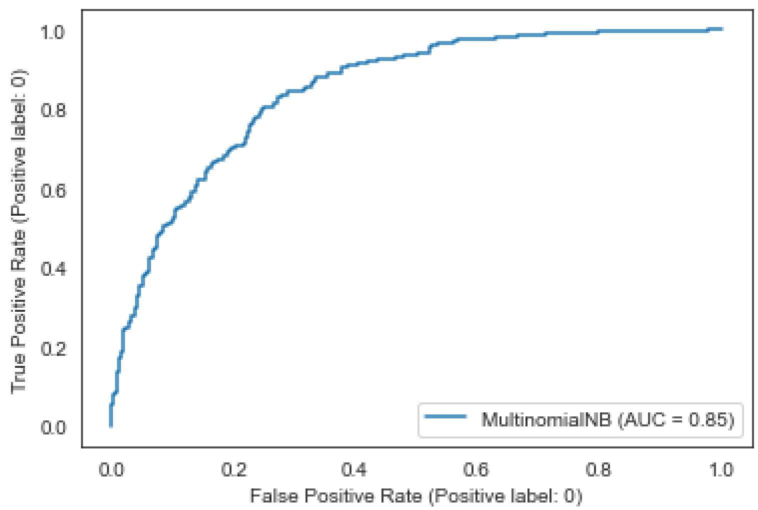


The Naive Bayes classifyer on CountVectorized training data was able to explain 96% of the variance, with the F1 score dropped about 6% from the AUC score at 90%, similar to the TFIDF metrics.

In [78]:
```
y_preds = nb_clf.predict(X_test_cv_vec.toarray())
evaluate(nb_clf, X_test_cv_vec.toarray(), y_test, y_preds)
```

executed in 165ms, finished 03:25:53 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd599215a90>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.77      0.70      0.73       329
Positive Sentiment       0.74      0.81      0.77       349

          accuracy                           0.75       678
         macro avg       0.76      0.75      0.75       678
      weighted avg       0.76      0.75      0.75       678


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



Simliar to the TFIDF experiments, the Naive Bayes algorithm had about a 10% drop in it's ability to explain the variance in data it had never seen before. We see the same healthy drop in F1 score by about 11%.

4.0.5 Cat Boost

Catboost expects either a DataFrame or an Array, but not a sparse matrix.

In [79]:
```
# instatiating the cat boost algorithm
cat_boost_clf = CatBoostClassifier(verbose = 0,
                                   loss_function = 'Logloss')
```
executed in 3ms, finished 03:25:53 2022-11-17

In [80]:
```
# making a catboost grid
cat_boost_grid = {'iterations': [5000, 6000, 7000],
 'depth': [3, 4, 5],
 'learning_rate': [0.01, 0.02, 0.03]}
```
executed in 2ms, finished 03:25:53 2022-11-17

TFIDF

In [81]:
```
#Applying classyfier and grid for the Gridsearch
cb_gs_tfidf = GridSearchCV(estimator = cat_boost_clf, param_grid = cat_boost_grid,
                           scoring = 'f1', cv = 5)

#Fitting model with the training data
cb_gs_tfidf.fit(X_train_tfidf_vec.toarray(), y_train)
```
executed in 5h 39m 28s, finished 09:05:21 2022-11-17

Out[81]:
```
▸ GridSearchCV
▸ estimator: CatBoostClassifier
  ▸ CatBoostClassifier
```

In [82]:
```
# investigating the best params
cb_gs_tfidf.best_params_
```
executed in 2ms, finished 09:05:21 2022-11-17

Out[82]:
```
{'depth': 5, 'iterations': 6000, 'learning_rate': 0.03}
```

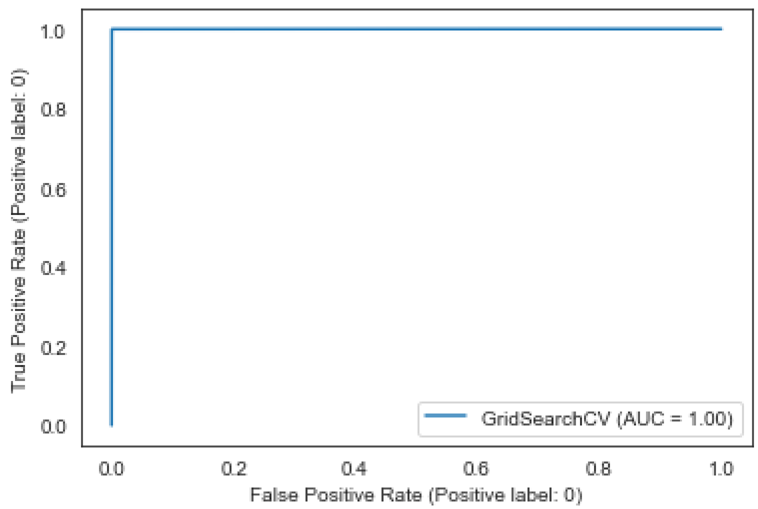
In [83]:
```
y_preds = get_cat_preds(cb_gs_tfidf, X_train_tfidf_vec.toarray())
```
executed in 15.8s, finished 09:05:37 2022-11-17

In [84]:
```
evaluate(cb_gs_tfidf, X_train_tfidf_vec.toarray(), y_train, y_preds)
```

executed in 15.9s, finished 09:05:53 2022-11-17

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd5993ebd60>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       1.00      1.00      1.00      2599
Positive Sentiment       1.00      1.00      1.00      2819

          accuracy                           1.00      5418
         macro avg       1.00      1.00      1.00      5418
      weighted avg       1.00      1.00      1.00      5418
```



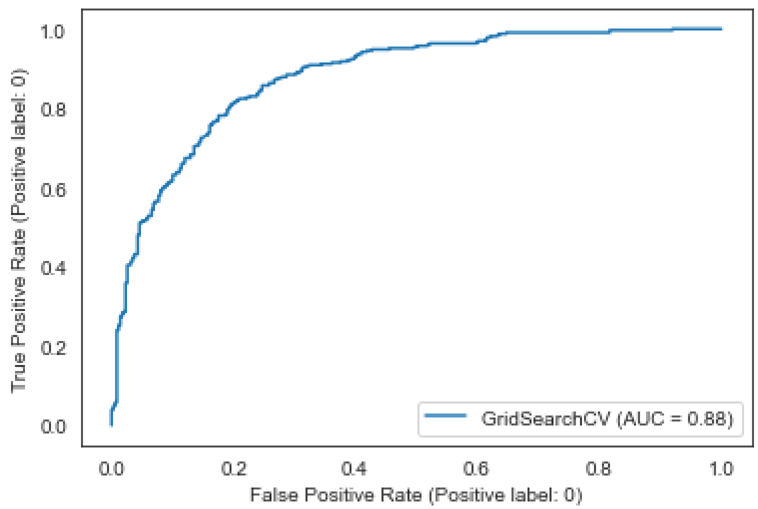
In [85]:
```
y_preds = get_cat_preds(cb_gs_tfidf, X_test_tfidf_vec.toarray())
```
executed in 1.83s, finished 09:05:55 2022-11-17

In [86]:

```
evaluate(cb_gs_tfidf, X_test_tfidf_vec.toarray(), y_test, y_preds)
```

executed in 1.94s, finished 09:05:57 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd5988e3160>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.80      0.79      0.80       329
Positive Sentiment       0.81      0.81      0.81       349

          accuracy                           0.80       678
         macro avg       0.80      0.80      0.80       678
      weighted avg       0.80      0.80      0.80       678


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



Count Vectorizer

In [88]:
```
#Applying pipeline and grid for the Gridsearch
cb_gs_cv = GridSearchCV(estimator = cat_boost_clf, param_grid = cat_boost_grid,
                        scoring = 'f1', cv = 5)

#Fitting model with the training data
cb_gs_cv.fit(X_train_cv_vec.toarray(), y_train)
```
executed in 4h 37m 27s, finished 15:29:48 2022-11-17

Out[88]:
```
▸ GridSearchCV
▸ estimator: CatBoostClassifier
  ▸ CatBoostClassifier
```

In [89]:
```
cb_gs_cv.best_params_
```
executed in 13ms, finished 17:32:44 2022-11-17

Out[89]:
```
{'depth': 4, 'iterations': 7000, 'learning_rate': 0.03}
```

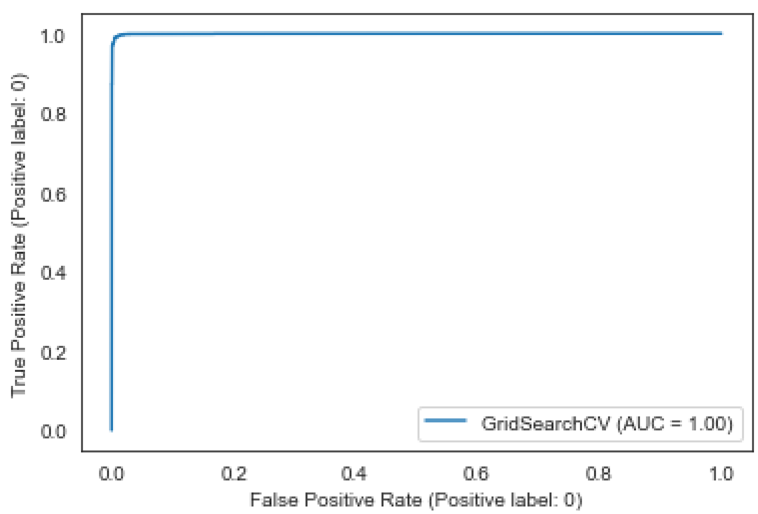
In [90]:
```
y_preds = get_cat_preds(cb_gs_cv, X_train_cv_vec.toarray())
```
executed in 16.0s, finished 17:33:04 2022-11-17

In [91]:

```
evaluate(cb_gs_cv, X_train_cv_vec.toarray(), y_train, y_preds)
```

executed in 16.0s, finished 17:33:22 2022-11-17

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)

<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd5e90b7d60>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.99      0.99      0.99      2599
Positive Sentiment       0.99      0.99      0.99      2819

          accuracy                           0.99      5418
         macro avg       0.99      0.99      0.99      5418
      weighted avg       0.99      0.99      0.99      5418
```



In [92]:

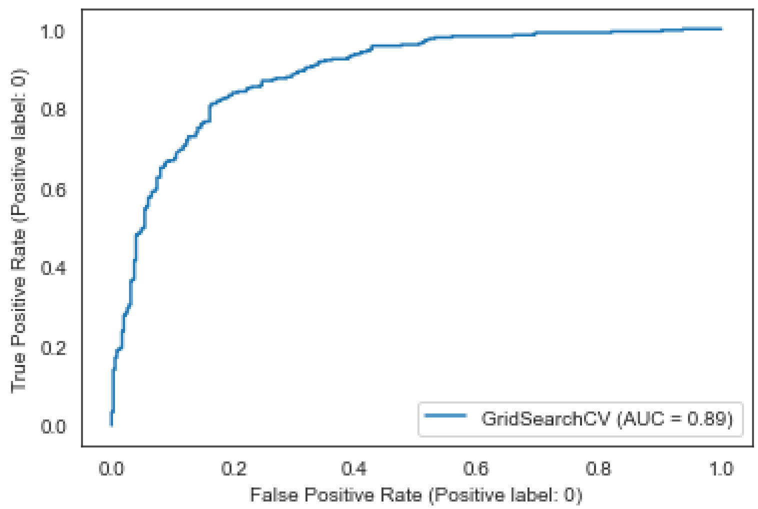
```
y_preds = get_cat_preds(cb_gs_cv, X_test_cv_vec.toarray())
```

executed in 1.79s, finished 17:33:32 2022-11-17

In [93]:
```
evaluate(cb_gs_cv, X_test_cv_vec.toarray(), y_test, y_preds)
```

executed in 1.91s, finished 17:33:35 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd598dc31f0>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.81      0.82      0.82       329
Positive Sentiment       0.83      0.82      0.82       349

          accuracy                           0.82       678
         macro avg       0.82      0.82      0.82       678
      weighted avg       0.82      0.82      0.82       678


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



5 Validation

In [94]:

```
# taking a look at the macro average precision score for each model

reports
```

executed in 26ms, finished 17:33:45 2022-11-17

```
  'support': 349},
 'accuracy': 0.8067846607669616,
 'macro avg': {'precision': 0.8071980558080984,
  'recall': 0.8060502869684116,
  'f1-score': 0.8063258430641416,
  'support': 678},
 'weighted avg': {'precision': 0.8070183188336911,
  'recall': 0.8067846607669616,
  'f1-score': 0.806603914399184,
  'support': 678}},
{'Negative Sentiment': {'precision': 0.9980754426481909,
  'recall': 0.9976914197768373,
  'f1-score': 0.9978833942659228,
  'support': 2599},
 'Positive Sentiment': {'precision': 0.997872340425532,
  'recall': 0.998226321390564,
  'f1-score': 0.9980492995211917,
  'support': 2819},
 'accuracy': 0.9979697305278701,
 'macro avg': {'precision': 0.9979738915368614,
```

In [95]:

```
# making reports into a df for ease of reading and sorting

precision_scores = []

for x in reports:
    for item in x.items():
        if 'macro avg' in item:
            precision_scores.append(item[1]['precision'])


names = ['Random Forest TFIDF train', 'Random Forest TFIDF test',
         'Random Forest CV train', 'Random Forest CV test',
        'XG Boost TFIDF train', 'XG Boost TFIDF test',
        'XG Boost CV train', 'XG Boost CV test',
        'Naive Bayes TFIDF train', 'Naive Bayes TFIDF test',
        'Naive Bayes CV train', 'Naive Bayes CV test',
        'CatBoost TFIDF train', 'Catboost TFIDF test',
        'Catboost CV train', 'Catboost CV test']

precision_scores = pd.DataFrame(list(zip(names, precision_scores)))
precision_scores.sort_values(by = [1])
```

executed in 25ms, finished 17:33:49 2022-11-17

Out[95]:

	0	1
9	Naive Bayes TFIDF test	0.752046
5	XG Boost TFIDF test	0.755162
11	Naive Bayes CV test	0.755643
7	XG Boost CV test	0.781722
13	Catboost TFIDF test	0.802216
1	Random Forest TFIDF test	0.807198
3	Random Forest CV test	0.812644
15	Catboost CV test	0.819898
6	XG Boost CV train	0.853884
4	XG Boost TFIDF train	0.868391

I'll be choosing the Catboost algorithm and using the CV data since that did the best (although marginally.)

In [96]:
```
best_params = cb_gs_cv.best_params_
best_params
```
executed in 9ms, finished 17:34:39 2022-11-17

Out[96]:
```
{'depth': 4, 'iterations': 7000, 'learning_rate': 0.03}
```

I'll be refitting a new catboost model with the data without using gridsearchCV, so that this model gets the full training dataset.

In [99]:
```
best_cat = CatBoostClassifier(depth = 4, iterations= 7000, learning_rate = 0.03,
                    verbose = 0, loss_function = 'Logloss')
best_cat.fit(X_train_cv_vec, y_train)
```
executed in 51.1s, finished 17:36:45 2022-11-17

Out[99]:
```
<catboost.core.CatBoostClassifier at 0x7fd599c5e6a0>
```

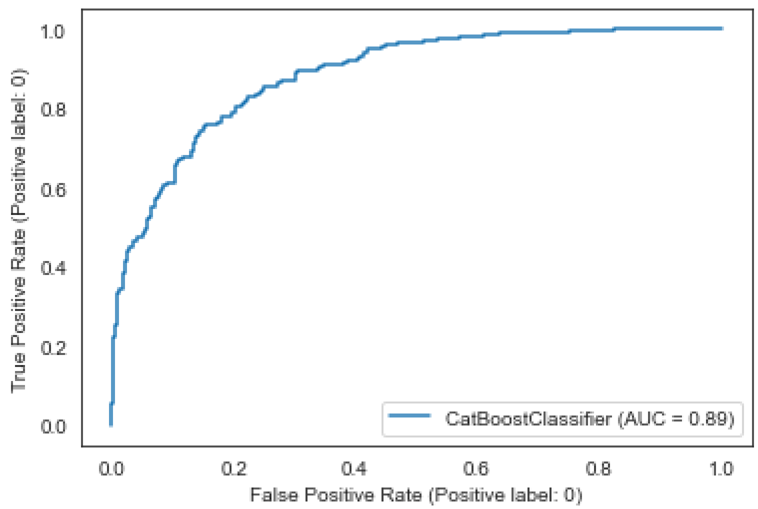
Now I'm going to evaluate the validation set which the model hasn't seen before

In [100]:

```
y_preds = best_cat.predict(X_valid_cv_vec)
evaluate(best_cat, X_valid_cv_vec, y_valid, y_preds)
```

executed in 197ms, finished 17:37:11 2022-11-17

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x7fd598a328e0>
************************************************************
FULL REPORT
************************************************************
                    precision    recall  f1-score   support

Negative Sentiment       0.80      0.76      0.78       328
Positive Sentiment       0.79      0.83      0.81       349

          accuracy                           0.79       677
         macro avg       0.80      0.79      0.79       677
      weighted avg       0.80      0.79      0.79       677


/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/utils/deprecation.p
y:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is depreca
ted in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```



observations: had the best macro average we've seen so far, at 80%. We do seem to have hit a threshold for how well our models can do given the limited data set and labeling system.

6 Saving the model

In [101]:
```
#saving the model to a pickle file

filename = 'hurrihelp_outreach_alorithm.sav'
pickle.dump(best_cat, open(filename, 'wb'))
```
executed in 55ms, finished 17:37:26 2022-11-17

7 Explainer with Lime

I'll be using Lime's text explainer to have a second look at our most common positive and negative words and see if we can't do further analysis.

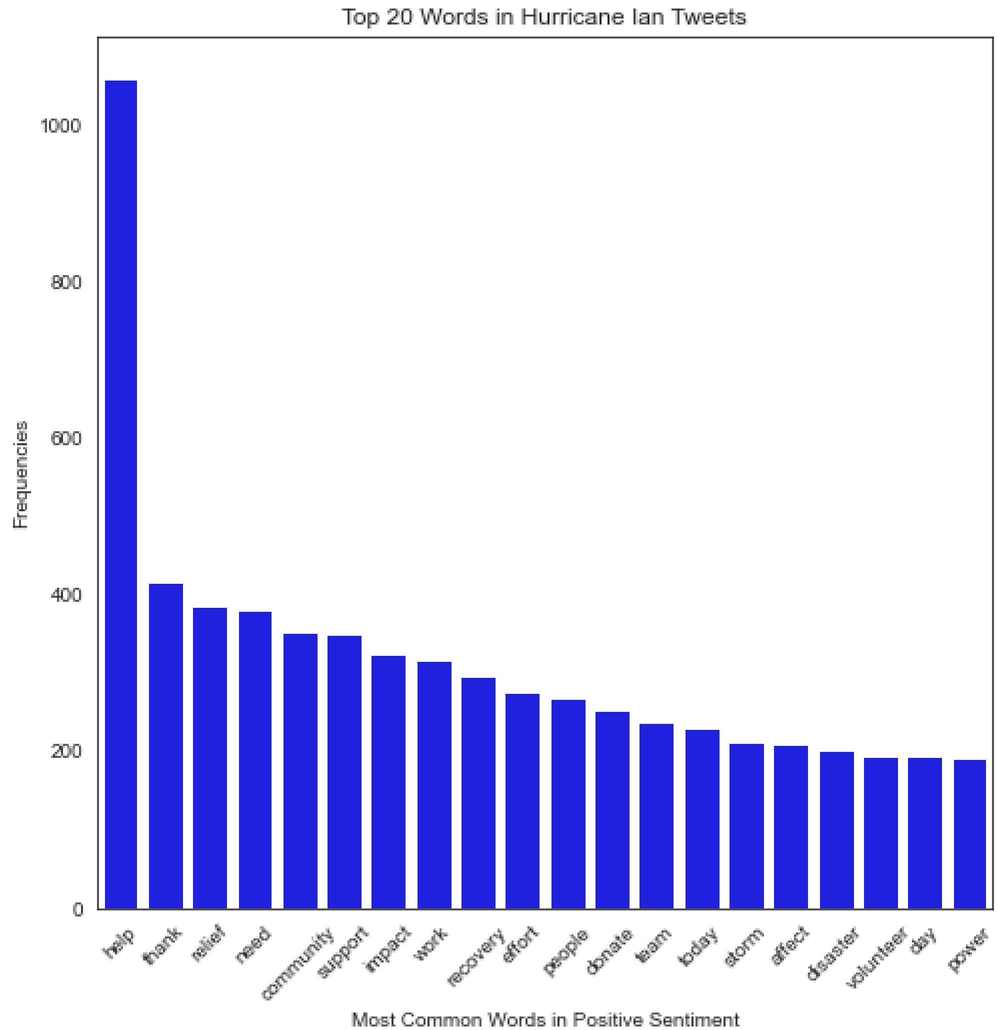
In [132]:
```
# making a pipeline for Lime to use

pipeline = make_pipeline(count_vectorizer, best_cat)
```
executed in 5ms, finished 17:48:56 2022-11-17

In [143]:
```
# making an explainer object

labels = ['Negative', 'Positive']
explainer = lime.lime_text.LimeTextExplainer(class_names = labels)
```
executed in 13ms, finished 18:37:21 2022-11-18

In [134]:

```
# refamiliarizing with most common positive words

most_common_posi_words = most_common(pos_doc_string, 'Positive Sentiment')[0]
```

executed in 6.50s, finished 17:49:07 2022-11-17



In [135]:
```
most_common_posi_words = [x[0] for x in most_common_posi_words]
most_common_posi_words
```
executed in 11ms, finished 17:49:09 2022-11-17

Out[135]:
```
['help',
 'thank',
 'relief',
 'need',
 'community',
 'support',
 'impact',
 'work',
 'recovery',
 'effort',
 'people',
 'donate',
 'team',
 'today',
 'storm',
 'affect',
 'disaster',
 'volunteer',
 'day',
 'power']
```

In [144]:

```
# making a loop to show the sentiment behind each of the most popular words in
# the positive dataset

for x in most_common_posi_words:
    exp = explainer.explain_instance(str(x), pipeline.predict_proba, num_features=50)
    exp.show_in_notebook(text=False)
```

executed in 3.00s, finished 18:37:35 2022-11-18


Prediction probabilities
Negative 0.59
Positive 0.41
Negative
Positive
impact
0.01


Prediction probabilities
Negative 0.56
Positive 0.44
Negative
Positive
work
0.02


Prediction probabilities
Negative 0.49
Positive 0.51
Negative
Positive
recovery
0.05


Prediction probabilities
Negative 0.42
Positive 0.58
Negative
Positive
effort
0.08


Prediction probabilities
Negative 0.72
Positive 0.28
Negative
Positive
people
0.04


Prediction probabilities
Negative 0.53
Positive 0.47
Negative
Positive
donate
0.03


Prediction probabilities
Negative 0.44
Positive 0.56
Negative
Positive
team
0.07

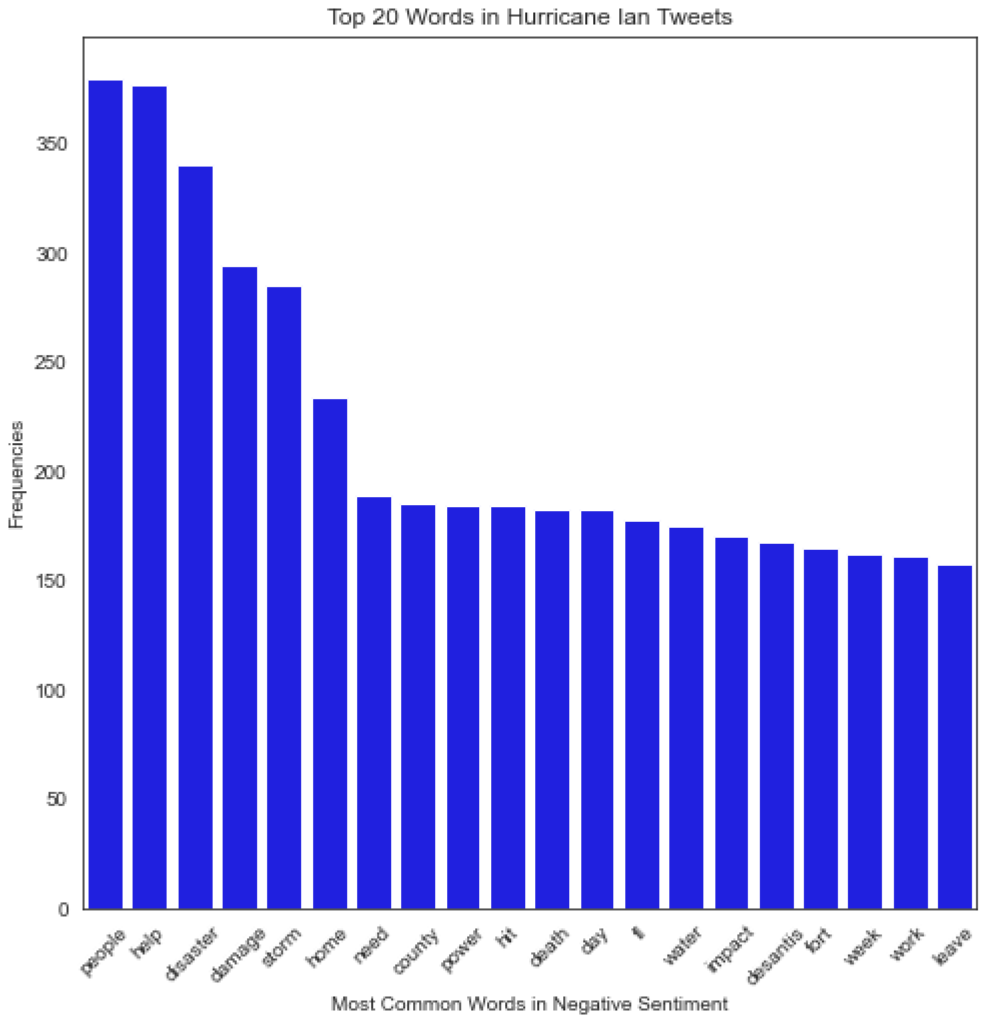

Negative
Positive
today
0.02

Observations: Disaster is the most negative word to be used often in tweets with positive sentiment. Also words: help, thanks, support and volenteer had the most positive sentiment attached out of all the words on this list. Those words can be used in the future to rule out tweets when we're looking for the target variable in the future.

In [141]:

```python
# Refamiliarizing with most common negative words

most_common_neg_words = most_common(neg_doc_string, 'Negative Sentiment')[0]
```

executed in 5.74s, finished 18:14:25 2022-11-18



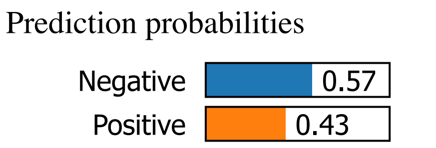
In [138]:
```
most_common_neg_words = [x[0] for x in most_common_neg_words]
most_common_neg_words
```
executed in 3ms, finished 17:49:18 2022-11-17

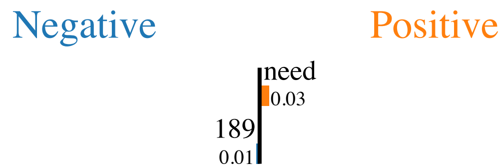
Out[138]:
```
['people',
 'help',
 'disaster',
 'damage',
 'storm',
 'home',
 'need',
 'county',
 'power',
 'hit',
 'death',
 'day',
 'fl',
 'water',
 'impact',
 'desantis',
 'fort',
 'week',
 'work',
 'leave']
```

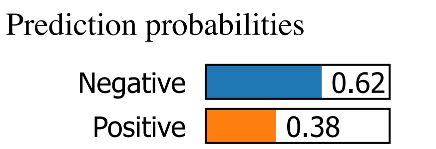
In [145]:

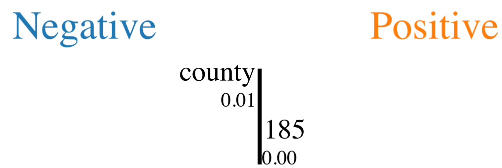
```
# explaining most negative words in dataset

for x in most_common_neg_words:
    exp = explainer.explain_instance(str(x), pipeline.predict_proba, num_features=50)
    exp.show_in_notebook(text=False)
```

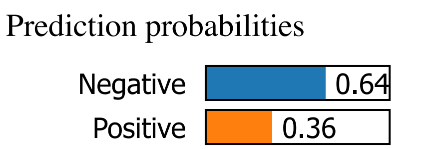
executed in 3.08s, finished 18:37:46 2022-11-18

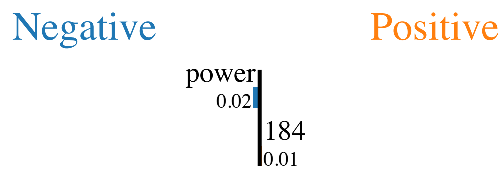

Prediction probabilities
Negative 0.57
Positive 0.43

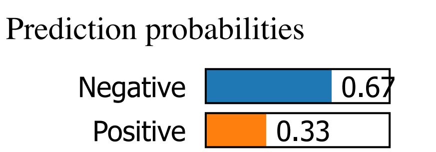

Negative
Positive
need
0.03
189
0.01

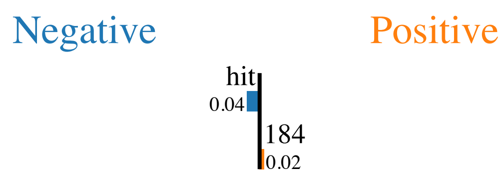

Prediction probabilities
Negative 0.62
Positive 0.38

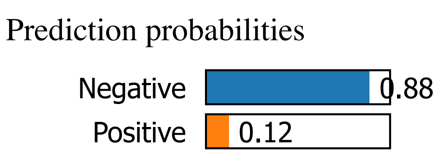

Negative
Positive
county
0.01
185
0.00

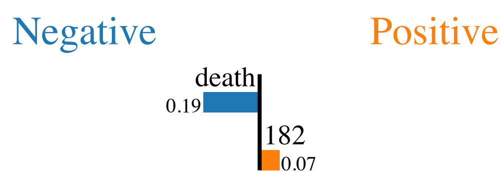

Prediction probabilities
Negative 0.64
Positive 0.36

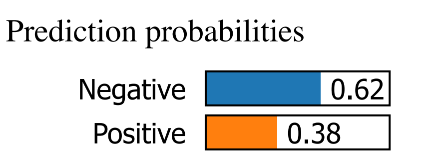

Negative
Positive
power
0.02
184
0.01

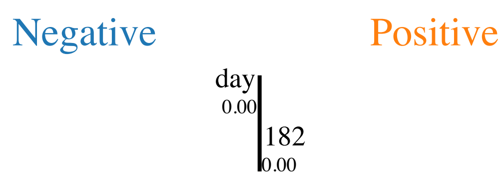

Prediction probabilities
Negative 0.67
Positive 0.33

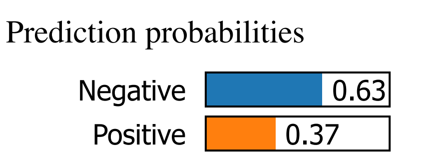

Negative
Positive
hit
0.04
184
0.02

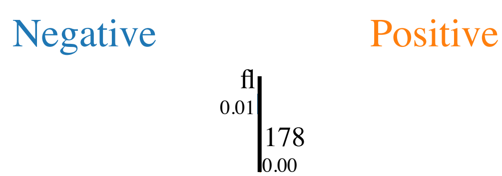

Prediction probabilities
Negative 0.88
Positive 0.12

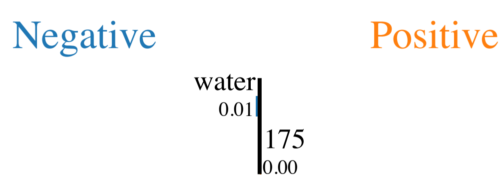

Negative
Positive
death
0.19
182
0.07


Prediction probabilities
Negative 0.62
Positive 0.38


Negative
Positive
day
0.00
182
0.00


Prediction probabilities
Negative 0.63
Positive 0.37


Negative
Positive
fl
0.01
178
0.00


Negative
Positive
water
0.01
175
0.00

Observations: 'help' is the most commonly appearing positive word in the most negative tweets. This is interesting because help, when talking about help given has a really positive sentiment, but when talking about needing help, that sentiment can also be negative.

8 Conclusion

As for negative words, the three Ds of Hurricane Ian are "Disaster", "Damage", and "Death" which were all the most negative in the top 20 most common words within the negative tweets. These words are indicators of a negative sentiment tweet.

Interesting find: It's interesting that DeSantis is on this list. Since tweets about DeSantis are so often having a negative sentiment, the algorithm currently does not differentiate angry tweets about the governor from desperate tweets of our target.

RECOMMENDATIONS to Hurricane Response:

Huge financial burden of recovery is common theme in negative tweets, more outreach by FEMA to inform public about financial options and disaster relief.

Look out for tweets with the following words: "Disaster", "Damage", and "Death" as these were the most singularly negative words in the data.

Future Work

Try EMOTION DETECTION algorithm to isolate 'SAD' tweets for labeling, and have labels as "TARGET" and "Not TARGET" for discernment.

Collect and utilize larger data set

Analyze and model other features in data set

Make pipeline for weeding out tweets and modeling tweets

Get a prototype of HurriHelp working live on twitter

Remove names and places from data so HurriHelp will be scalable for future hurricanes

Most Importantly: PROJECT NEEDS A NEW HOME

In [ ]:

```

```