TABLE OF CONTENTS

Problem Definition:

With the emergence of new technologies, our ability to communicate with others has increased dramatically. No longer can communities rely solely on buying traditional media, newspapers, radio, and TV news. With the emergence of smartphone technology, communities are simply an 'app' away from being able to deliver or receive information within milliseconds.

Recently, Twitter has been used to spread updates/news all over the world and has been effective in emergencies during natural calamities like floods, earthquakes and wildfires. Twitter has proved that it has the potential to increase survival of a person during a Tornado related disaster. Communication forums allow for the communication of various networks that you can help officials during a disaster to make a list of victims, deceased, and contact with family and friends of all victims during the communication and planning of victims and respondents.

Twitter has become an invaluable tool throughout the world, especially for spreading and discussing issues in everyday life. There are an estimated 500 million Twitter users worldwide. But the tweet generated is not always informative in nature. Thus there should be some method to classify the tweet.

Twitter has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. In times of emergencies there will be a lot of tweets being floating on twitter. But it's not always clear whether a person's words are actually announcing a disaster. It might be clear to a human eye right away but less clear to a machine. We build a machine learning model that predicts which tweets are real about disasters and which one's are not.

Study and Understanding of Algorithm:

Dataset:

In this study we use the social media disaster tweets dataset provided by Kaggle. It is freely downloadable from https://www.kaggle.com/c/nlp-getting-started/data.

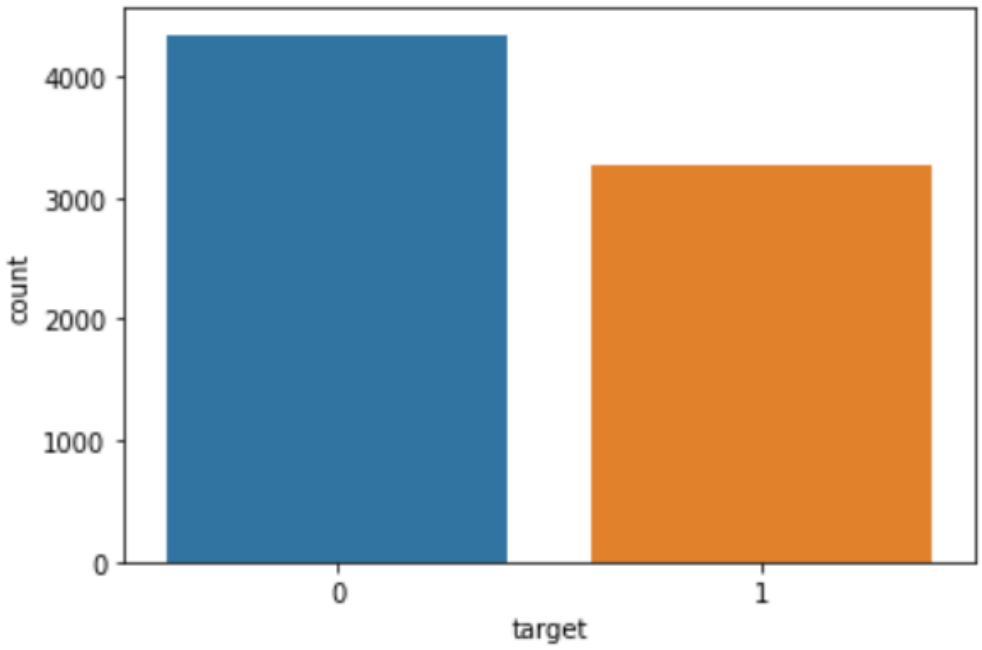
In this dataset they have provided more than 10,000 comments and we estimate whether a particular tweet is about a specific catastrophe or not.

Each entry in dataset includes following details:

1. Id – tweet identifier
2. Text - text of the tweet
3. Location – location from where the tweet was sent
4. Keyword – A relevant keyword in the tweet
5. Target – Whether tweet is a real disaster(1) or not(0)

Distribution:

```
Target of 0 is 57 % of total
Target of 1 is 43 % of total
```



Preprocessing:

1. Data Cleaning
2. Make text all lower or uppercase
3. Removing Noise
4. Tokenization
5. Stop word Removal
6. Stemming
7. Lemmatization

Before pre-processing:

```
#Lets have a quick look of the text data
train_df['text'][:5]
```

```
0    Our Deeds are the Reason of this #earthquake M...
1               Forest fire near La Ronge Sask. Canada
2    All residents asked to 'shelter in place' are ...
3    13,000 people receive #wildfires evacuation or...
4    Just got sent this photo from Ruby #Alaska as ...
Name: text, dtype: object
```

After pre-processing:

```
# Text after pre-processing the text column
train_df2.head()
```

	id	keyword	text	target	text_length
0	1	NaN	deeds reason earthquake may allah forgive us	1	69
1	4	NaN	forest fire near la ronge sask canada	1	38
2	5	NaN	residents asked shelter place notified officer...	1	133
3	6	NaN	people receive wildfires evacuation orders cal...	1	65
4	7	NaN	got sent photo ruby alaska smoke wildfires pou...	1	88

Vectorization of pre-processed text:

1. Bag of Words
2. TFIDF Features

Model Performance Metrics:

1. Confusion Matrix
2. F1-Score
3. Accuracy

Models:

We have implemented the following models

1. Logistic Regression
2. DecisionTreeClassifier
3. RandomForestClassifier
4. BERT

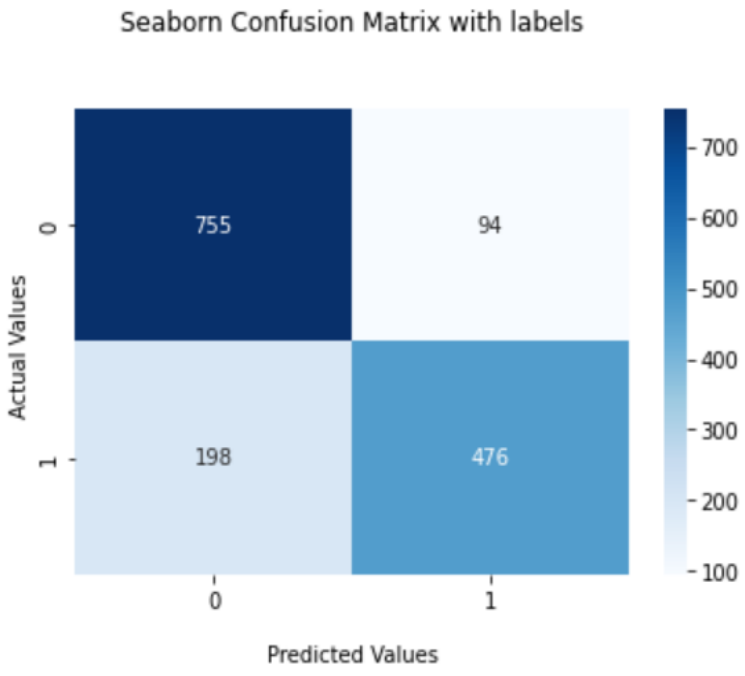
Experimentation and Results:

Experimented these models with above mentioned vectorizers and reported the results below.

Logistic Regression:

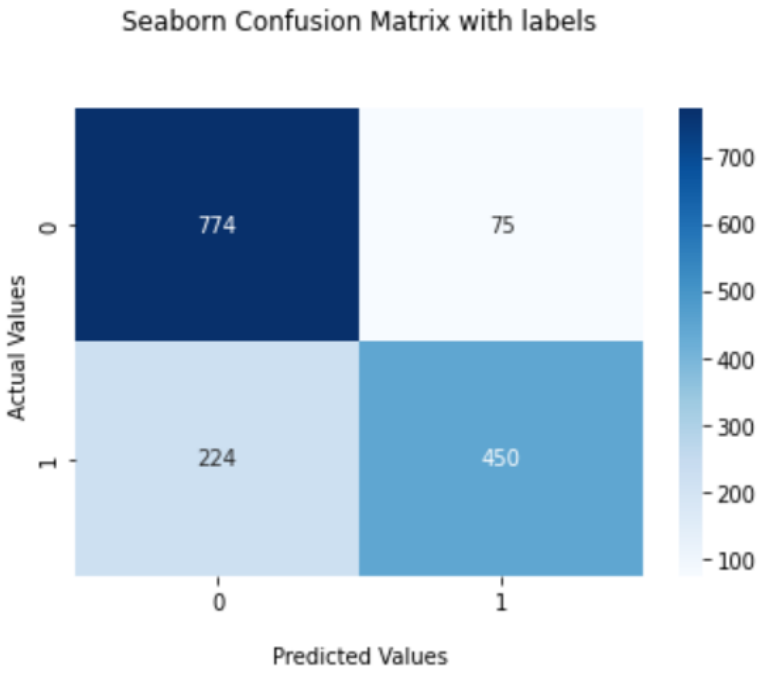
Bag of words:

Accuracy: 0.81
F1 Score: 0.81



TFIDF Features:

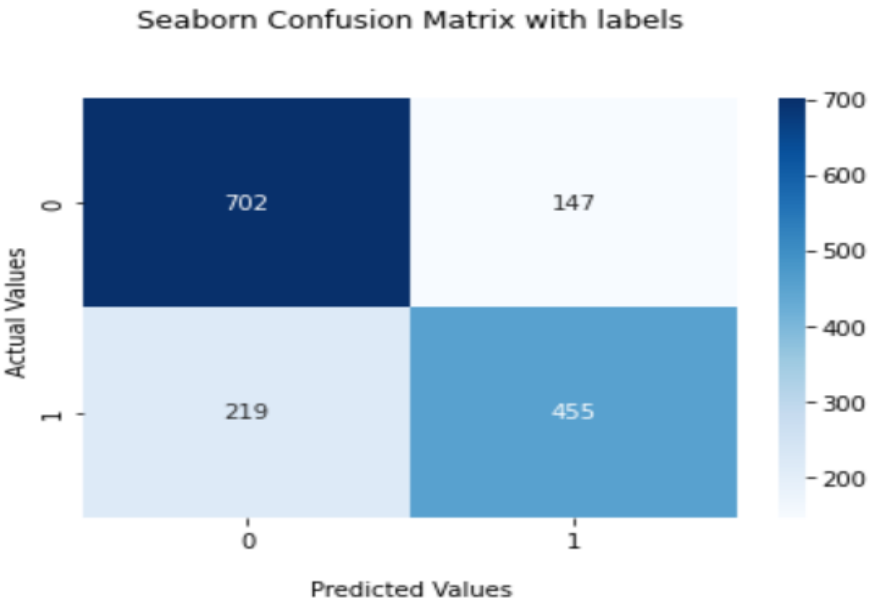
Accuracy: 0.80
F1 Score: 0.80



Decision Tree Classifier:

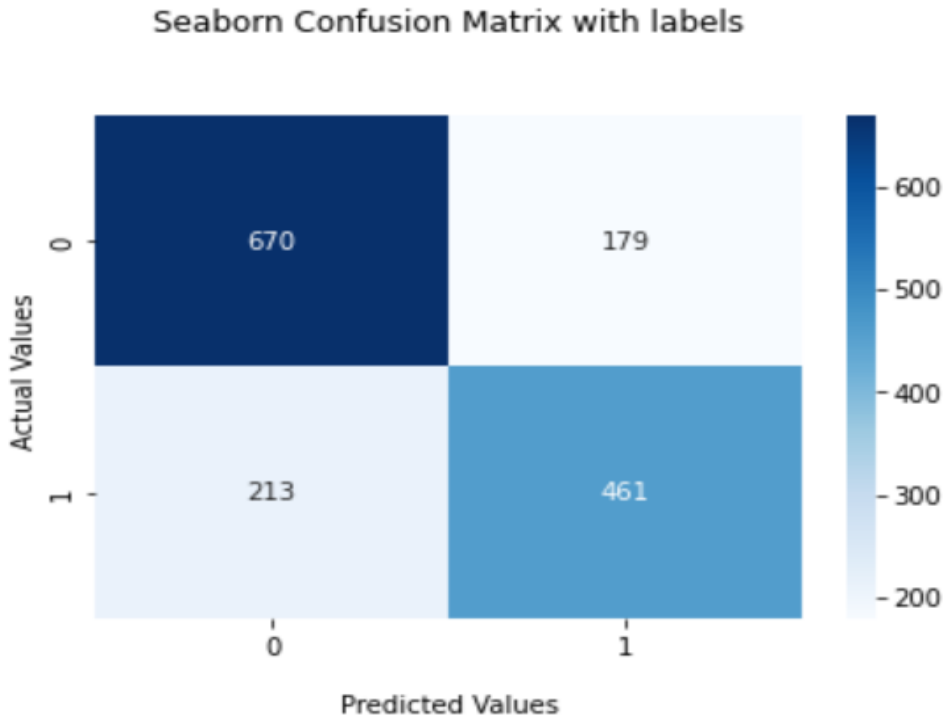
Bag of Words:

Accuracy: 0.76
F1Score: 0.76



TFIDF Features:

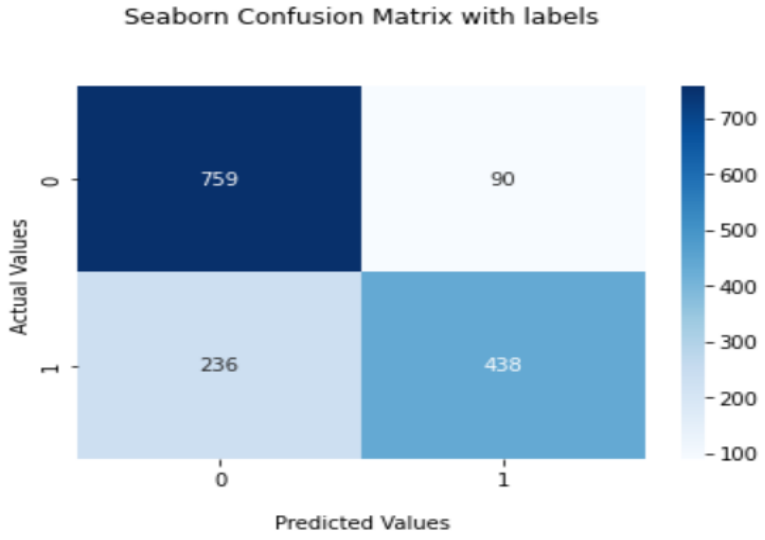
Accuracy: 0.74
F1Score: 0.74



Random Forest Classifier:

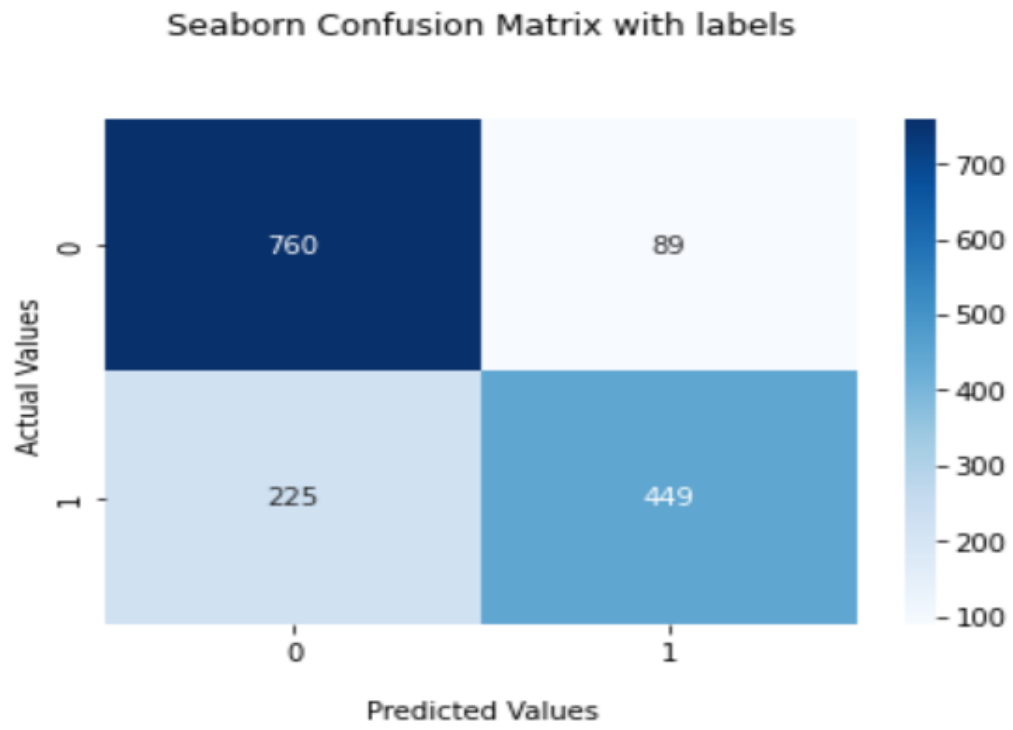
Bag of Words

Accuracy: 0.79
F1Score: 0.78



TFIDF Features

Accuracy: 0.79
F1Score: 0.79



BERT:

```
[ ] model.fit(X_train, y_train, epochs=5)

    Epoch 1/5
    191/191 [==============================] - 1907s 10s/step - loss: 0.6270 - accuracy: 0.6517 - precision: 0.6414 - recall: 0.4299
    Epoch 2/5
    191/191 [==============================] - 1902s 10s/step - loss: 0.5753 - accuracy: 0.7179 - precision: 0.7211 - recall: 0.5602
    Epoch 3/5
    191/191 [==============================] - 1908s 10s/step - loss: 0.5565 - accuracy: 0.7312 - precision: 0.7281 - recall: 0.5976
    Epoch 4/5
    191/191 [==============================] - 1905s 10s/step - loss: 0.5459 - accuracy: 0.7394 - precision: 0.7365 - recall: 0.6129
    Epoch 5/5
    191/191 [==============================] - 1900s 10s/step - loss: 0.5388 - accuracy: 0.7442 - precision: 0.7393 - recall: 0.6251
    <keras.callbacks.History at 0x7f478cc097d0>
```

Architecture:

```
Model: "model"
__________________________________________________________________________________________
 Layer (type)                Output Shape              Param #
Connected to
==========================================================================================
 text (InputLayer)           [(None,)]                 0           []

 keras_layer (KerasLayer)    {'input_type_ids':   0
['text[0][0]']
                             (None, 128),
                              'input_word_ids':
                             (None, 128),
                              'input_mask': (Non
                             e, 128)}

 keras_layer_1 (KerasLayer)  {'default': (None,   109482241
['keras_layer[0][0]',
                             768),
'keras_layer[0][1]',
                              'pooled_output': (
'keras_layer[0][2]']
                             None, 768),
                              'encoder_outputs':
                              [(None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768)],
                              'sequence_output':
                              (None, 128, 768)}

 dropout (Dropout)           (None, 768)               0
['keras_layer_1[0][13]']

 output (Dense)              (None, 1)                 769
['dropout[0][0]']

==========================================================================================
Total params: 109,483,010
Trainable params: 769
Non-trainable params: 109,482,241
```

Conclusion:

Considering the very large number of tweets data over the internet, it is a diligent task to classify them. Because of the presence of non-textual data in tweets and the unique feature of a particular tweet, text classification has become a challenging task. This system propose a tweet classifier framework which takes in input as raw tweet and outputs are classifications of the tweet as informative or non-informative. By using this machine learning model we can classify the tweets. This Project can be extended to classification of multi-model posts by training with additional dataets.

References:

1. https://www.researchgate.net/publication/350716764_Disaster_Tweets_Classification_in_Disaster_Response_using_Bidirectional_Encoder_Representations_from_Transformer_BERT

2. https://www.researchgate.net/publication/350716764_Disaster_Tweets_Classification_in_Disaster_Response_using_Bidirectional_Encoder_Representations_from_Transformer_BERT