Ad ML
Group 9

Proposal

Topic discussion: NLP with disaster tweets

https://www.kaggle.com/c/nlp-getting-started/overview

backup:

- House price prediction

https://www.kaggle.com/c/house-prices-advanced-regression-techniques

- Volcanic eruption prediction

https://www.kaggle.com/c/predict-volcanic-eruptions-ingv-oe/overview

- TREC-COVID(NLP)

https://www.kaggle.com/c/trec-covid-information-retrieval/overview(might need deep learning idk)

- Netflix Recommendations (past data)

https://www.kaggle.com/netflix-inc/netflix-prize-data

- Amazon Products Reviewing by Customers (past data)

https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products

- Helping fill job vacancies in Los Angeles, CA (competition)

https://www.kaggle.com/c/data-science-for-good-city-of-los-angeles

Project Proposal

Group 9 Raymond Luo, Xinyue Ji, Xingying Wang, Jiahao Wu, Dakota Schey

Topic Predict which Tweets are about real disasters by Natural Language Processing

Objective of the analysis

Living under the influence of global pandemics, social media has become one of the most frequently used and powerful mediums for people to communicate and share important information. When major disasters happen, most people will choose social media platforms, such as Twitter, Instagram, and Snapchat, as their ways of doing disaster podcasts, raising funds, and any other content related to disasters. Hence, based on the above information, our group presumes that disaster relief organizations, news agencies, and local governments will have a demand of monitoring public opinions posted on these social media so that they can respond to disasters promptly, alleviate the situation, and reduce the losses caused by the disaster.

Under this circumstance, our group wants to build an NLP model using approximately 10,000 pieces of text data in Twitter and forecast what piece of information or which post is disaster-related. Moreover, these text data are derived from an ongoing Kaggle competition, and the link of the competition is included at the end of this proposal. Also, using data from a Kaggle competition enables us to test the validity and efficiency of our NLP model. We can modify our model whenever the final prediction is not satisfying.

Finally, we decided to further our research by obtaining real-time Twitter posts and utilizing our NLP model to find any content related to disasters.

Dataset https://www.kaggle.com/c/nlp-getting-started/overview

Necessary items:

Likely to use Sentiment Analysis

Ex. to adopt code from in terms of sentiment analysis, setting “1” as our target variable for disaster-related tweets, and “0” as our target variable for non-disaster-related tweets

Process:

- Import Necessary Dependencies (top of python file)
- Read and Load the Dataset (still have to set dummy variables for USA/non-USA location)
- Exploratory Data Analysis
- Data Visualization of Target Variables
- Data Preprocessing
    - Remove @’s
    - Remove other special characters (!?,etc.)
    - Remove unnecessary words (<=2 letters)
    - Tokenization
    - Stemming vs. Lemmatization? (started w/ Stemming but Lemmatization might be better)
- Splitting our data into Train and Test Subset (done already)
- Transforming Dataset using TF-IDF Vectorizer
- Function for Model Evaluation
- Model Building
- Conclusion

https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/

Project Report

1. Objective

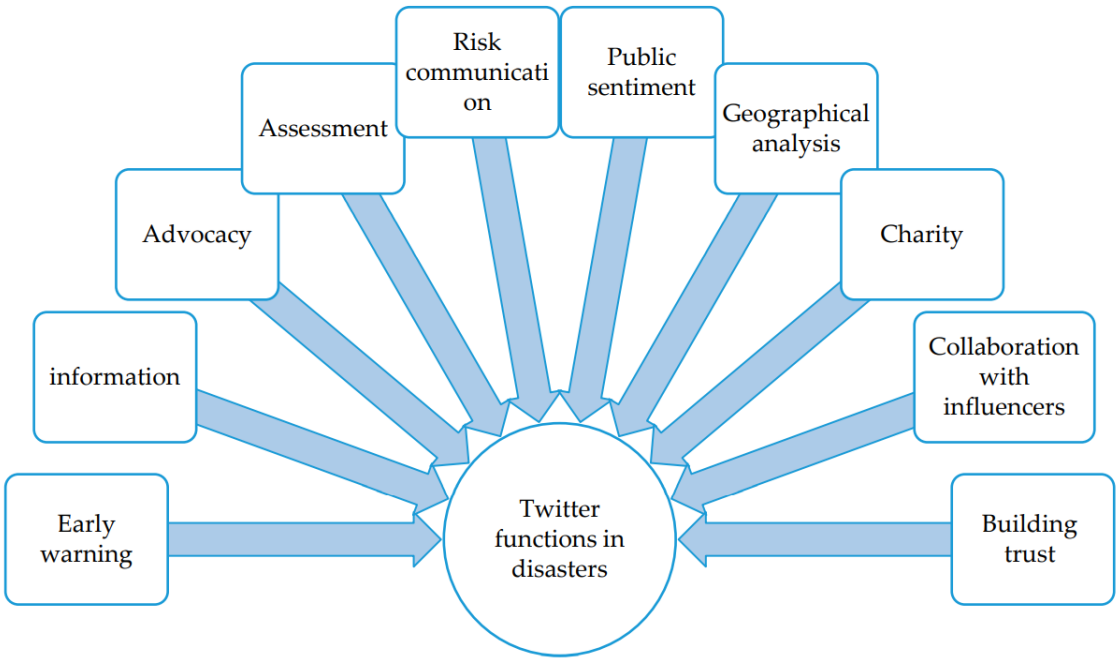
The ultimate objective of the project is to predict selected disasters through analysis of social media posts by utilizing NLP and machine learning techniques. The prediction primarily includes the geographical location of the disaster and specific time to achieve emergency management activities, so that rescue activities could be in place in a timely way.

Living under the influence of global pandemics, social media has become one of the most frequently used and powerful mediums for people to communicate and share important information. When major disasters happen, most people will choose social media platforms, such as Twitter, Instagram, and Snapchat, as their ways of doing disaster podcasts, raising funds, and any other content related to disasters. Hence, based on the above information, our group presumes that disaster relief organizations, news agencies, and local governments will have a demand of monitoring public opinions posted on these social media so that they can respond to disasters promptly, alleviate the situation, and reduce the losses caused by the disaster.

2. Background

As the internet and technology evolves, social media plays an important role in society and posting posts, specifically called “microblogging”, has become one of the primary ways for people to social and network with others. With the development of machine learning and deep learning, the value of microblogging has evolved from personal networking tools to create value for the whole society, such as detecting disasters and emergency rescue management. Through analysis of people’s postings about what happened around them, disasters could be detected more timely with supplies provided sooner. In this way, the harm brought by the disaster could be reduced and less people may suffer from the pain. At the same time, it results in a lower cost for the country to recover.

Twitter has become an important communication channel in times of emergency. Portable devices enable people to announce an emergency they’re observing in real-time. Because of this, more agencies are interested in programatically monitoring Twitter.



This figure indicates ten functions of Twitter in disasters, including early warning, dissemination of information, advocacy, assessment, risk communication, public sentiment, geographical analysis, charity, collaboration with influencers and building trust.

Preventing the spread of misinformation is one of the most important issues in times of disaster, especially pandemics. Sharing accurate, transparent and prompt information from emergency organizations and governments can help. Moreover, analyzing Twitter data can be a good way to understand the mental state of the community, estimate the number of injured people, estimate the points affected by disasters and model the prevalence of epidemics. Therefore, various groups such as politicians, government, nongovernmental organizations, aid workers and the health system can use this information to plan and implement interventions.

3. Scope

For one tweet, it may include various types of information such as image, text, location, user id, and so on. Our analysis only covers the text data of tweets. This is because:

In terms of the location, it is manually entered by the user, so we may have a lot of missing values. What's more, the accuracy of the user's input cannot be guaranteed. Therefore, the location is not regarded as a feature.

In terms of the image, Kaggle did not provide us with disaster image data. Therefore, we need to manually collect pictures and label them before we train the model, which is really time-consuming. In addition, the way to process images is quite different from text data. Therefore, images are not regarded as a feature.

4. Data

Our group wants to build an NLP model using approximately 10,000 pieces of text data in Twitter, which serves to forecast what piece of information or which post is disaster-related.

The training dataset is derived from an ongoing Kaggle competition. We will further divide the dataset into training and validation set to fine-tune hyperparameters.

Dataset https://www.kaggle.com/c/nlp-getting-started/overview

As seen below, the DataFrame contains 4 feature and 1 label column:

- **id:** a unique identifier of every tweet
- **keyword:** a particular keyword from the tweet (this can be blank)
- **location:** the location the tweet was sent from (this can be blank)
- **text:** the text of the tweet

- **target (label):** present only in the train data, and denotes if the tweet is about a real disaster (1) or not (0)

For the testing dataset,we plan to use web crawler to grab data from twitter directly. After preprocessing, we obtained 18,000 text data to make predictions on them. The testing set is made up of id, keyword, and text.

5. Data Analysis and Model Building Result

Data Background

In terms of the training set that we have before us, we need to understand the features and nuances of our dataset. To begin, our training set (provided by Kaggle) has about 7,600 different data points all positioned under 4 features, with the inclusion of one target variable: “target”. We split this initial training set into both our training and validation set. Our testing set has over 22,000 different data points, all located under the realm of each feature. To understand our variables, we’ll list their meanings and definitions below:

Features:
id: Numerical number assigned to each individual tweet as a way of uniquely identifying each tweet
text: The text as seen from the tweet
location: The location where the tweet was sent from
keyword: Particular keyword that is part of a tweet
Clean_tweet_num (additional input added)**:** Custom column created to calculate how many characters exist in each tweet
Outcome:
target: Denotes whether a tweet is about a natural disaster (1) or not about a natural disaster (0) [train only]

For the purposes of our experiment, we can neglect the “id” and ”location” variables in terms of our training data, mainly focusing on the “text” and “keyword” variables and utilizing “clean_tweet_num” to help with data preprocessing.

Data Analysis - Training Set Overview

Text & Tweet Analysis

Taking a quick look at our main feature, “text”, we can see that there are a jumble of different grammatical structures that each of these tweets can have. Just in terms of words, we have over 28,000 unique words appearing throughout the entire dataset of tweets. Characters, as well, such as @’s, “&’s”, “;” and mismatched capitalization structures seem to be very prevalent in our training dataset.

text	clean_tweet_num
'Burning Rahm': Let's hope City Hall builds a ...	143
Schoolboy ÛÒ Aftershock (Original Mix)\nExcis...	149
@rewind_music found out about you guys today(r...	147
so privileged and proud to wear this uniform.?...	141

As well, we have to be cognizant of the custom column “clean_tweet_num”, which is primarily used to understand and analyze tweet lengths. Because Twitter has a maximum Tweet length of 140 characters (as of now it’s actually 280, but initially it was designed to be 140), we have to discard any tweets with lengths of more than 140 characters.

Data Preprocessing

An important aspect of NLP Machine Learning is to make sure our tweets are cleansed in a sense that the algorithm can make use of very simplified tweets in order to make accurate predictions. What this entails is:

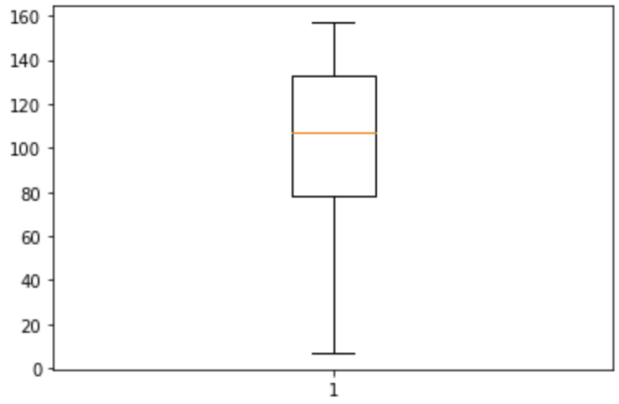
- Removing URLS, Emojis, URL tags, punctuation and STOPWORDS
- Stemming & Lemmatizing the data to receive the base root of a word (ie. running->run)
- Lower-casing the text data
- Tokenizing & Vectorizing the data to apply directly into NLP model

Why Tokenize and Vectorize?

Machines can't decipher the text of a tweet without applying some sort of tokenization as a way of providing "numerical" context to the meaning of a word. By doing this, we can change the word "our" to the code "2256", or an input that a computer model can actually understand and learn from.

We apply TF-IDF Vectorization as a way of measuring the originality of the words appearing in our tweets. This is achieved by determining the frequency of a particular word in a document as well as the frequency of the number of documents the particular word appears in. By doing this, we can penalize or reward certain words based on the "weight" that these words have in affecting the meaning of our sentences.

Tweet Specifics



Boxplot depicting the length of tweets that appear in our training dataset

As we can see from the boxplot above, the majority of our tweets range in length from about 80-130 characters, with the median falling right around 110 characters.

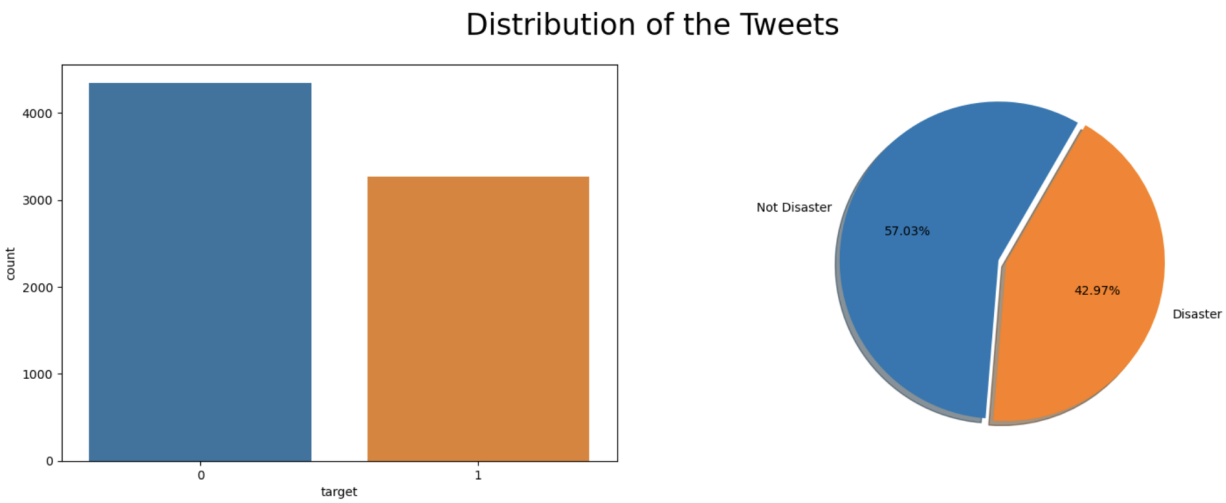
Emoticons

Emojis, another element that needs to be considered in our training set, can be defined in dictionary form with all of their corresponding meanings. When we visualize all of the tweets that exist in our training dataset we get:

```
[(1843, ': '),
 (19, ':)'),
 (16, 'xv'),
 (15, 'XD'),
 (12, 'x2'),
 (11, 'Xo'),
 (9, 'XL'),
 (9, ':('),
 (7, ':-)'),
 (6, ';)'),
 (5, ':S'),
 (4, '::'),
 (3, ';-;'),
 (1, 'X:'),
 (1, ';-)')]
```

From this, we can see that the first emoji appeared 1,843 times throughout our text column. These all have to be removed in order to help the machine algorithm properly analyze the sentiment of our tweets.

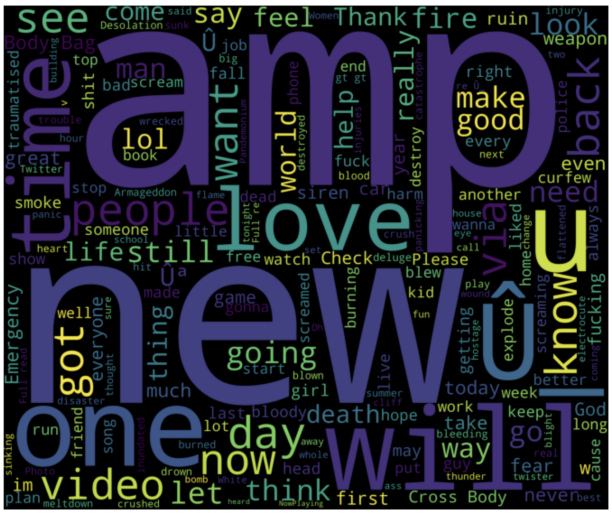
Target Analysis - what Exactly is our Training Outcome?



As evident by our visuals, our target variable had about 14% more non-disaster than disaster tweets

Comparative Insights on our Training Set Tweets: Disaster vs. Non-Disaster Tweets

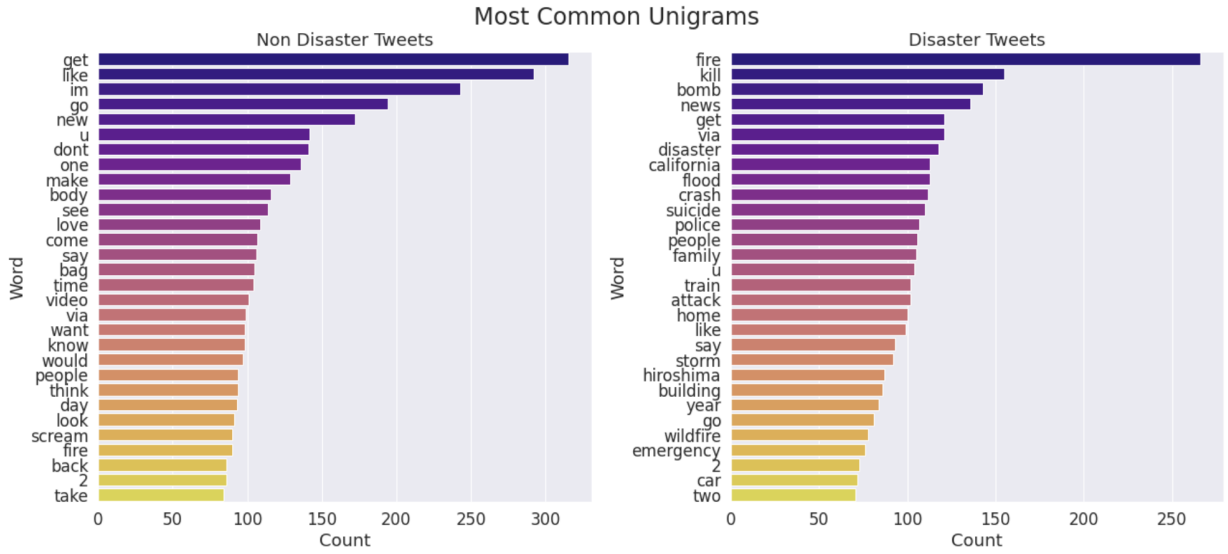
When we think of creating a model that depicts disaster from non-disaster tweets, what factors would one expect to look at? Maybe the sentiment of each tweet is a good place to start, and to do this, we can explicitly look at the most common words that appear in the categorized disaster and non-disaster tweets



Most common words in non-disaster tweets

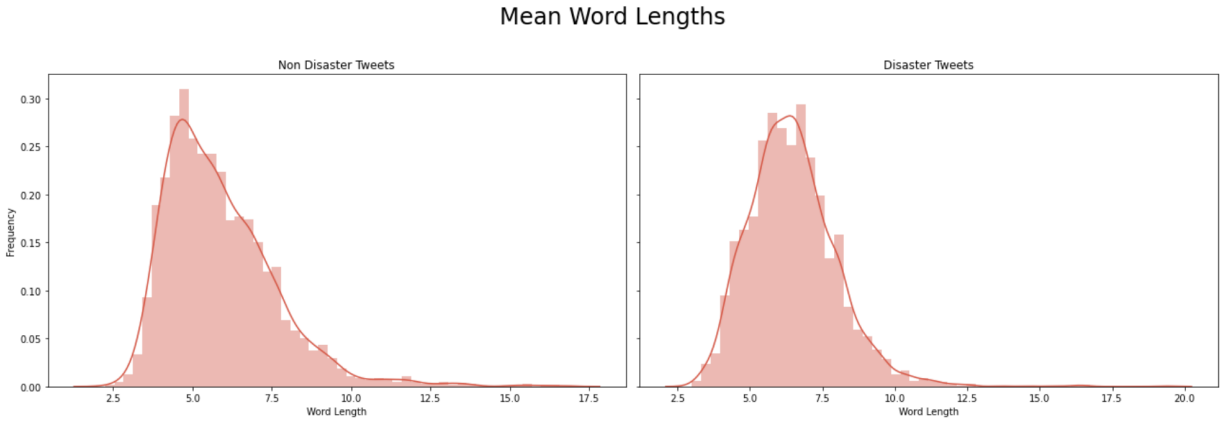
Most common words in disaster tweets

Based on the WordCloud images above, we can see that generic-sounding words such as “love”, “amp” and “new” make up the majority of words that appear in non-disaster tweets. On the other hand, very disaster-esque words appear in the disaster-categorized tweets. We clearly see the words “fire”, “Hiroshima” and “storm” appear before our eyes in the right picture. These words are much more indicative of a disaster-related theme, explaining why such a model can make predictions based on the sentiment of a sentence diction.



Another important distinction between disaster and non-disaster tweets has to do with the most common unigrams under each category. For example, the most common unigrams for non-disaster tweets are largely stopwords, while the most common unigrams for disaster tweets are incredibly relevant to common crisis or alarm-raising words that one would expect to encounter when dealing with a disaster. It's rather interesting to see how clear of a distinction exists between disaster and non-disaster tweets, I guess making it easier for the model to learn.

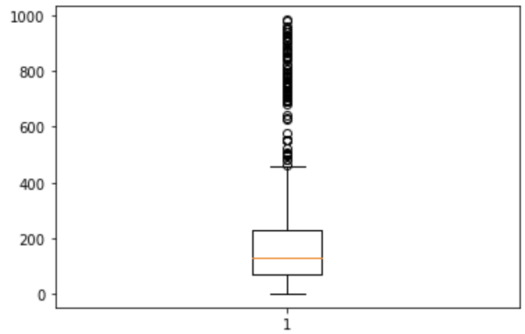
In terms of mean character and mean word length of disaster-oriented and non-disaster oriented tweets, we could see that disaster tweets had a higher frequency of more words and characters appearing in such tweets and words. This is likely due to the supportive background text and "substance" that often exists to provide some context around the obscurity of a disaster. For example, when a fire occurs somewhere, more information is often supplied alongside the context of the event to give readers more clarity about the situation happening and the severity of its consequences. An example of this may be something alongside the lines of: "A *ravaging* fire occurred in *Santa Barbara, California"*. The italicized words are examples of "padding" terms that can be used to add clarity in a crisis-ridden situation.



Mean word length hovers consistently around 6 characters for disaster related tweets, while it hovers around 5 characters and falls off rather rapidly in a right-skewed distribution for non-disaster tweets - this can be attributed to shorter-character words often making up the abundance of non-disaster tweet length ('get', 'like', 'love', etc.)

A Quick Visualization: Text Data

Our text data, because we are using it for prediction purposes, does not contain a "Target" column now to tell us which one of our tweets is Disaster or Non-Disaster based. This is what we hope our training model can help clarify for us after fine-tuning the algorithm. However, as of now, we can still do a little bit of initial data analysis to understand the text data that we're working with.



The boxplot above depicts the spread of tweet character length for our testing set. As we can see, the majority of character lengths for tweets falls between about 80-200, which is slightly larger than the average tweet character length of our training set. The median of tweet character length for our testing set falls right around 120-140 characters.

Most Common Words (Alphabetically)

```
There is 52789 different words

['ablaze',
 'able',
 'absolutely',
 'accident',
 'accidents',
 'account',
 'across',
 'act',
 'action',
 'actually',
```

Taking a quick glance at our “different words” chart above, we can see that the testing set contains about 2.5 times as many unique words than the training set does. This chart above DOES NOT tell the word frequency, but it is ordered in alphabetical order.

Emoticons

```
[(8244, ':/'),
 (1826, ': '),
 (1548, '; '),
 (990, 'xp'),
 (93, ':3'),
 (57, 'xd'),
 (44, 'xs'),
 (42, 'Xi'),
 (42, ':)'),
 (41, 'xD'),
 (39, ':('),
 (37, 'x5'),
 (34, 'XD'),
 (32, 'xl'),
 (23, 'X5'),
 (22, 'Xq'),
 (22, ';;'),
 (21, 'xq'),
 (15, ':D'),
 (14, ';)'),
 (13, ':-'),
 (10, '::'),
 (4, 'X,'),
 (4, ';-;'),
 (4, ':P'),
 (4, ":')"),
 (3, ':L'),
 (2, ':p'),
 (1, ':V')]
```

To begin, we can already see that our testing set twitter text data contains a good amount more unique emojis through its entirety. While there are only about 10 more unique emojis that appear throughout our text dataset, we also have to be cognizant of the frequency of each emoji appearing. For example, the “:/” appears more times throughout the testing set than there are rows of data in our training set! While this won’t likely cause issues, we should definitely have some familiarity with the textual data we are planning to predict.

Model Building

Model Selection

- KNN
- MLP
- SVM
- Stacking

Model Implementation

- KNN

For implementing the K-Nearest-Neighbors (Knn) algorithm, we used the KNeighborsClassifier model in the sklearn library. The grid_search_knn() function will test all combinations of parameters within the parameters dictionary defined within the function using the GridSearchCV model found in the sklearn library, and will run combinations in parallel. This function will return the best parameters for implementing the Knn algorithm. The knn() function is a standalone implementation of the kNN model that takes the training data, testing data, and model parameters to use as the input parameters. This function will also produce a ROC curve as we choose ROC & AUC as the performance metric. For our implementation, only the k-nearest-neighbor value is used in the Knn model as a parameter for both the standalone and the grid searching. From our grid searching, we tested k-nearest-neighbor values ranging from 1 to 50, and found that k equal to 3 provided the best results. The model had the option of changing various other parameters, but from our testing, leaving everything else to default produced the best results. The comparison of this model's performance with other implementations is shown in the next section. This Knn model produced an AUC value of 0.67, which was relatively poor.

- MLP

For the multilayer perceptron(MLP) algorithm, we used the MLPClassifier model from sklearn library. There are several parameters in the MLPClassifier and we tested several combinations of these parameters to see the best result. The set of parameters we changed included alpha and hidden layer size. After the testing, we found that with alpha equaled to 1e-4 and hidden layer size of (6,4), we got the best AUC value (0.67). We also plotted the ROC curve in the below section for model comparison. The MLP model ended up with an AUC value of 0.79, which was better than the Knn model.

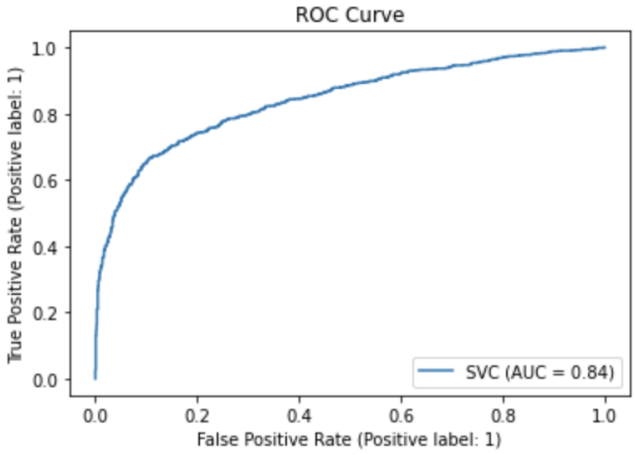
- SVM

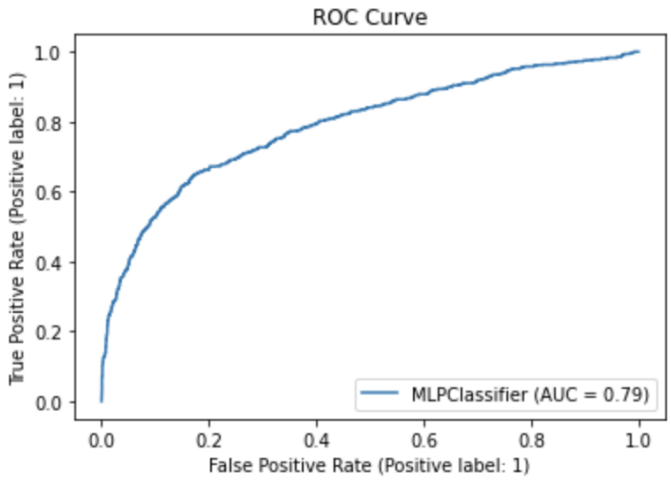
For implementing the Support Vector Machine (SVM) algorithm, we used the SVM model in the sklearn library. The svm() function is a standalone implementation of the SVM model that takes the training data and testing data to use as the input parameters. This function will also produce a ROC curve. The SVM model had the option of changing various parameters, but from our testing, leaving everything to default produced the best results. Lastly, this SVM model produced an AUC value of 0.84, making it the most effective model among the three.

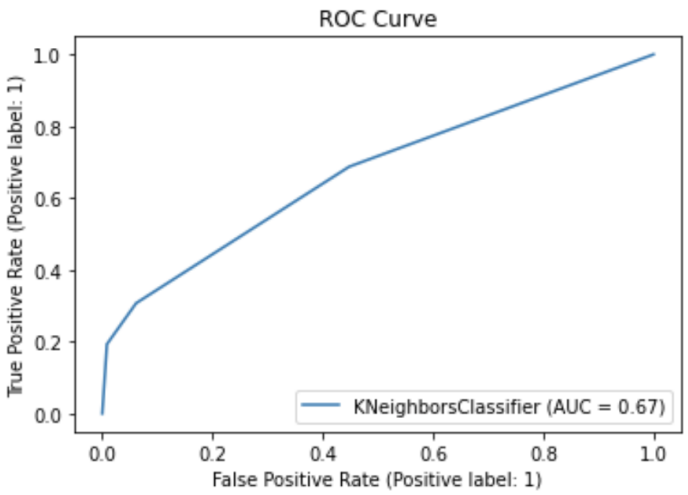
- Stacking Model

Besides the three models mentioned above, we also tried the stacking model to see if it could produce the best result. We combined the MLP, SVM and logistic regression models but the result wasn't ideal as it only produced an accuracy rate of 78%, which wasn't better than the standalone SVM model we tested.

Model evaluation







6. Key insights with impact

As is discussed in the background part, Twitter is becoming a real-time platform to track events, including accidents, disasters, and emergencies, especially in the era of mobile Internet and 5G communication. Hence, agencies like disaster relief organizations and news agencies are deploying resources to programmatically monitor Twitter.

The machine learning model we built exactly serves this purpose. In practice, these agencies shall monitor tweets in real time, perform text classification on each Tweet to predict disasters and emergencies with our model. If the prediction result shows that a certain tweet is disaster-related, the system alarm shall be triggered and provided to the manual for secondary review and confirmation.

By this process, disaster relief organizations and news agencies can greatly increase their reaction speed. As timing is the most critical factor, with our model, first responders can be dispatched and rescue plans can be made at the earliest time if disaster really happens. Moreover, if these agencies further monitor the location and release time of the disaster tweets, they could rely on algorithms to determine the location of people who need help.

Certainly, we acknowledge that the model we built can be optimized, in terms of the prediction accuracy and the real function. For instance, we may further try using machine learning methods to sort the tweets by their urgency, so that these agencies are able to prioritize the most urgent disaster needs. This will be discussed in the next part.

7. Recommendations（N/A）

***not applicable**

8. Next Steps

Overall, our model performs well but we find some problems that we should pay attention to and there is still a lot we can improve.

Weakness:

Data:

- **Limited Data**

We only have a limited volume of training dataset, which kind of limits the model performance. To improve the classification accuracy, we may need more text data.

- **Dirty Scraped Data**

Even if we set the language to be English when scrapping the data, tweets in multiple languages were included while our training dataset is in English. Some tweets were scrapped because the user name contained the keywords while the text did not, bringing lots of information unrelated to disasters. Besides, we are unable to remove all the typos, curse words, and retweets, etc. When we make the final prediction on scraped data, these noises may impact the performance of our model. We should improve our work in data cleaning.

Model:

The keyword column in the training dataset is not used at all in our existing model. Wecan try more models, like BERT, to see if it can increase the classification accuracy.

Future Thoughts:

Include other features:

We found that Many tweets are vague but posted with pictures. In these tweets, users usually used short phrases or single words to describe the pictures. Only dealing with text information, the model may not fully get the real meaning. We could introduce CNN to process images. Besides, location and time can also be included, because they are very important in disaster early warning, risk communication, geographical analysis etc. in real life.

Multiple platforms:

Our model is based on data in twitter text. While in reality, Snapchat, instagram and other social media are also platforms that should be monitored. And different social media may present different user behavior patterns due to user segments. We can do some analysis on different platforms to get some insights about how people in different age groups, etc react in disasters differ.

_____NOT USED____

- Process data:
    - Import Necessary Dependencies (top of python file)
    - Read and Load the Dataset
    - Exploratory Data Analysis
    - Data Visualization of Target Variables
    - Data Preprocessing
        - Remove @’s
        - Remove other special characters (!?,etc.)
        - Remove unnecessary words (<=2 letters)
        - Tokenization
        - Stemming vs. Lemmatization? (started w/ Stemming but Lemmatization might be better)
    - Splitting our data into Train and Test Subset
    - Transforming Dataset using TF-IDF Vectorizer to create pipeline
- Splitting our Data into train-test-validation sets
- Utilizing different models to test our disaster tweet ‘sentiment analysis’ problem