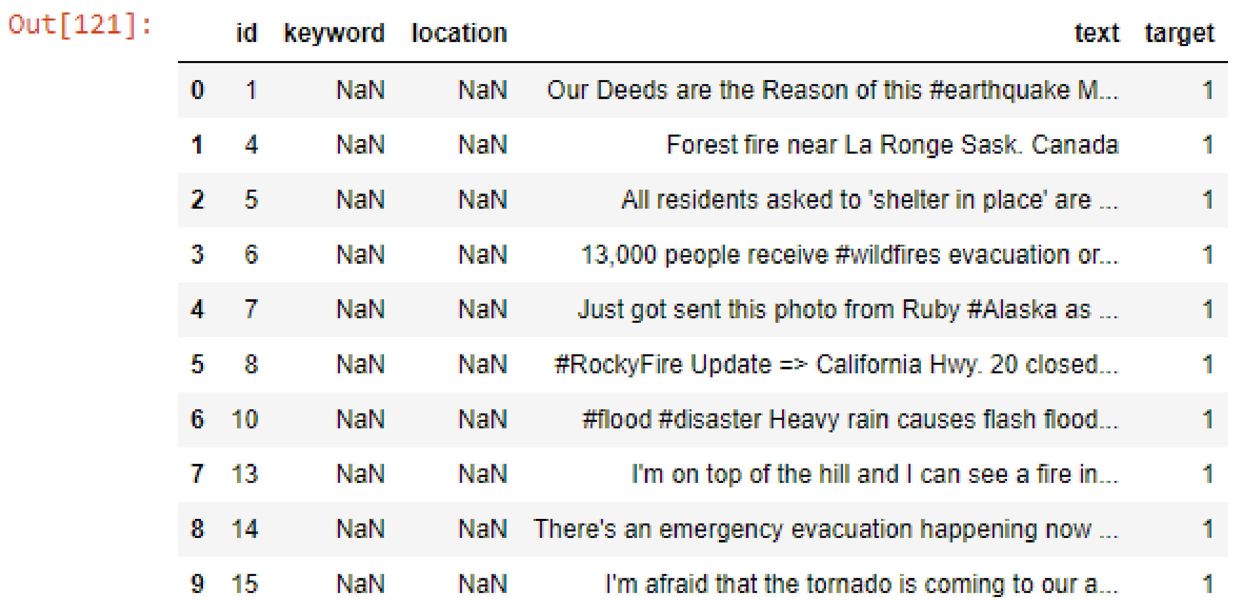
Processing with Disaster Tweets
(Predict which Tweets are about real disasters and which one are not)

AthriNandan Goud Enuganti and Roshan Sah
UID: U01037532 and U01022933
{enuganti.3, sah.5}@wright.edu

Introduction:

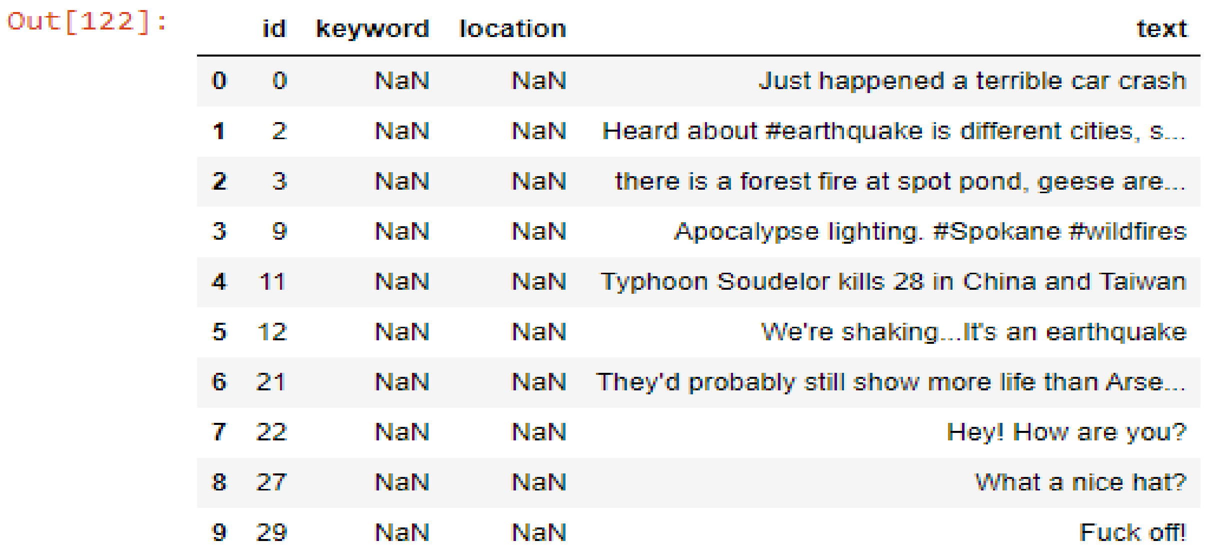
Twitter is an open social media platform which is used in this modern era where users of different parts of region may share their views, opinions, news, and other sorts of information that may be of interest to others. Where some of the information may be disaster-related and non-disaster-related, and people believe it and share it, causing others to fear. But it's not always clear whether a person's tweets are really announcing disaster or not. For instance, In the recent panoramic period, many people spread news about how Covid-19 is spreading, where some of them are true and some of them are real disaster.
Primarily focus of our work is to classify tweet data into real disaster and non-disaster classes. Since, Working with Twitter data has become an integral part of sentiment analysis problems therefore we will analyze the spread of disaster speech on twitter. We are utilizing the training dataset that consists of 7613 rows and 5 columns and testing dataset that consist of 3253 rows and 4 columns provided by kaggel.com. The sample of training dataset and testing dataset is shown in figure 1 and figure 2 respectively.



Out[121]:

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1
5	8	NaN	NaN	#RockyFire Update => California Hwy. 20 closed...	1
6	10	NaN	NaN	#flood #disaster Heavy rain causes flash flood...	1
7	13	NaN	NaN	I'm on top of the hill and I can see a fire in...	1
8	14	NaN	NaN	There's an emergency evacuation happening now ...	1
9	15	NaN	NaN	I'm afraid that the tornado is coming to our a...	1

Figure 1: Train Data Set



```
Out[122]:
```

	id	keyword	location	text
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan
5	12	NaN	NaN	We're shaking...It's an earthquake
6	21	NaN	NaN	They'd probably still show more life than Arse...
7	22	NaN	NaN	Hey! How are you?
8	27	NaN	NaN	What a nice hat?
9	29	NaN	NaN	Fuck off!

Figure 2: Test Data Set

As shown in the figure 3, we will adopt the following steps for the implementation of our project:
Step 1: Data collection: Twitter data in .csv format has been collected from Kaggel.com.
Step 2: Exploratory Data analysis: We will perform Data analysis on textual data to know about the distribution of tweets according to disaster and non-disaster classes. That will help us to deal with imbalanced or balanced dataset.
Step 3: Preprocessing: In our first iteration we performed the data preprocessing for the cleaning of data and represents the tweets in meaningful way to train the model. Detailed steps of preprocessing have been cited in Methods section of this document.
Step 4: Model Implementation: To find out the best suitable classification model we will going to utilize 4 machine learning model for the classification of disaster and non-disaster tweets.
Step 5: Results: For the evaluation purpose we will utilize confusion matrices for each Model.
Step 6: Conclusion: Further we will showcase the comparison of implemented classification models and conclude the best suitable Model.

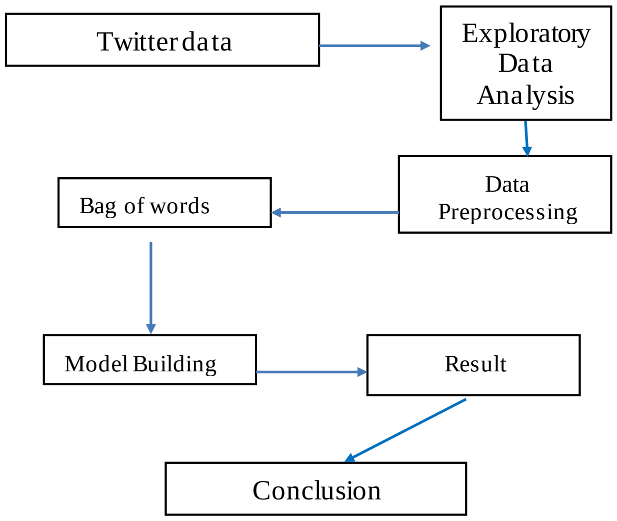


Figure 3: Illustration of Approach

Following are the important Python libraries that we have used:

- Nltk
- Numpy
- scikit-learns
- matplotlib
- seaborn
- pandas
- re

Exploratory Data Analysis (EDA):

The dataset is in csv file of type are:

- id: A unique identifier for each tweet.
- keyword: A particular keyword from the tweet (may be blank).
- location: The location the tweet was sent from (may be blank).
- text: The text of the tweet.
- target: This denotes whether a tweet is about a real disaster (1) or not (0).

The table 1 and figure 4 illustrates the total number of training dataset with respect to disaster and non-disaster tweets.

Tweets	**Total**	**Target**
Disaster	**3271**	**1**
Non-Disaster	**4342**	**0**

Table 1: Total Training dataset

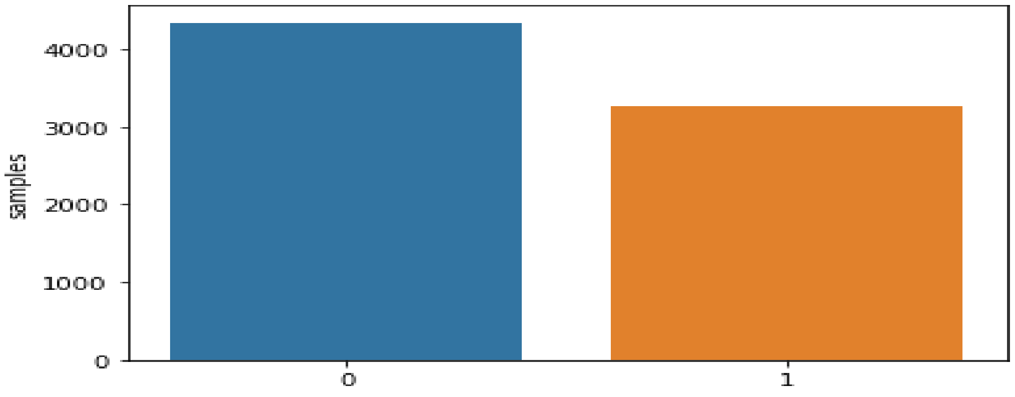
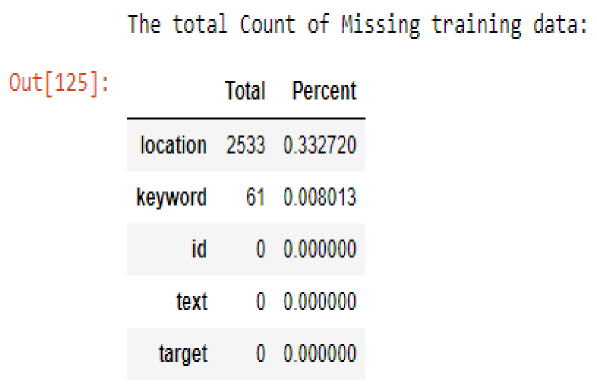


Figure 4: Target values of disaster and Non-disaster

Next,we are going to count the amount of data that's missing from each of the features in both train and test dataset which is shown in figure 5 and 6 respectively.
From the figures, we can see that most of the location are missing from the both training and

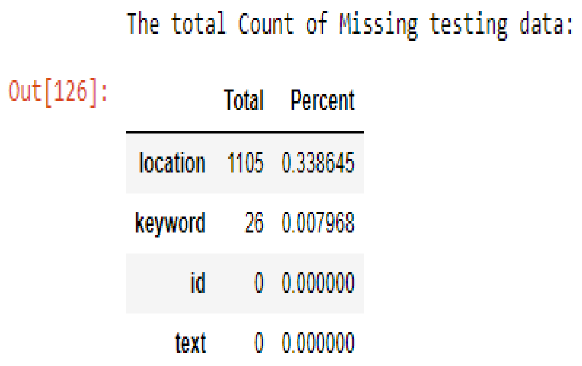
testing data set which are 0.3327% form the training and 0.3386% from the testing data set. But in the case of keyword features, only few are missing from the both training and testing data set. There are 221 unique Keywords for both training and testing dataset and 3341 unique location variables for the training dataset and1602 unique location variables in testing dataset.



```
The total Count of Missing training data:
Out[125]:
```

	Total	Percent
location	2533	0.332720
keyword	61	0.008013
id	0	0.000000
text	0	0.000000
target	0	0.000000

Figure 5: Count of Missing training data

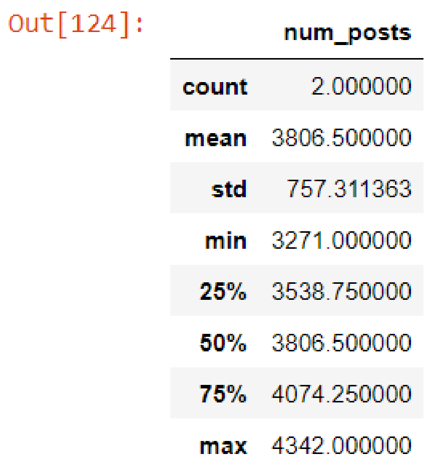


```
The total Count of Missing testing data:
Out[126]:
```

	Total	Percent
location	1105	0.338645
keyword	26	0.007968
id	0	0.000000
text	0	0.000000

Figure 6: Count of Missing testing data

The table 2 illustrates the statistics of the tweets of the training dataset which is based on the target of the dataset (0 or 1).



```
Out[124]:
```

	num_posts
count	2.000000
mean	3806.500000
std	757.311363
min	3271.000000
25%	3538.750000
50%	3806.500000
75%	4074.250000
max	4342.000000

Table 2: Descriptive statistics of tweets

Methods:

1. **Data Pre-processing**
   Since our data set is textual data that requires extensive data preprocessing. For the pre-processing, we use only the twitter text parameters. To fulfill the requirement of efficient training of model we performed following preprocessing steps:

a. **Unnecessary words removal**

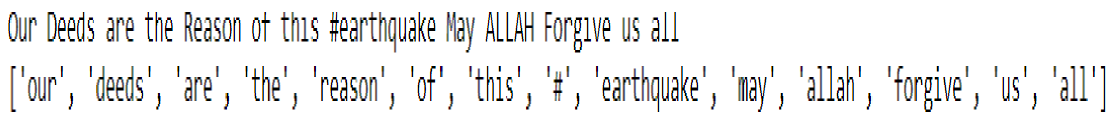
We remove the unnecessary words like symbols, emoji, hyperlink and URL. These words will make the model imperfect and our desired output may will not come. So, we remove these words.

b. **Lowering the entire text**

We transform words to lowercase because each letter has own ASCII Code that represent text in computers, Uppercase letter has different ASCII Code than same letter in lowercase format. so that **'A'** letter differs from **'a'** letter in computer.

c. **Tokenization of sentences**

We tokenize the pre-processed tweets. There are a variety of ways to do this, but we chose to use the Tweet Tokenizer from NLTK. It knows to keep emojis together as a token and to keep hashtags together. We already removed all the Twitter handles and the tokenized sentence is shown in the figure 7 as compared with the untokenized twitter text.

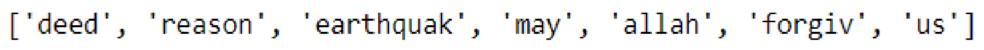


```
Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all
['our', 'deeds', 'are', 'the', 'reason', 'of', 'this', '#', 'earthquake', 'may', 'allah', 'forgive', 'us', 'all']
```

Figure 7: tweet text with the tokenized

d. **Stemming of words**

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. There are several types of stemming algorithms available and one of the famous one is porter stemmer which is widely used. We can use nltk package for the same. From the figure 8, we can illustrate those words like *earthquake* and *forgive* have their *e* at the end chopped off due to stemming. This is not intended, what can we do for that, but we can use Lemmatization in such cases.

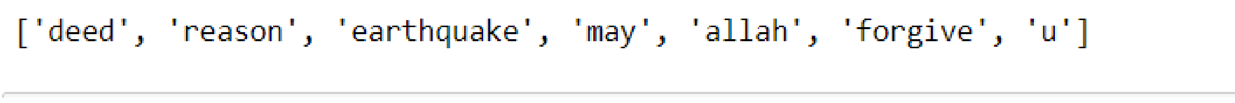


```
['deed', 'reason', 'earthquak', 'may', 'allah', 'forgiv', 'us']
```

Figure 8: Stemmed Words

e. **Lemmatization of words**

Lemmatization is used for reducing inflected words to their word stem and makes sure the root word (also called as lemma) belongs to the language. We use the Word Net Lemmatizer in nltk to lemmatize our sentences. Fromm the figure 9, we can see that the trailing *e* in the *earthquake* and *forgive* is retained when we use lemmatization unlike stemming.

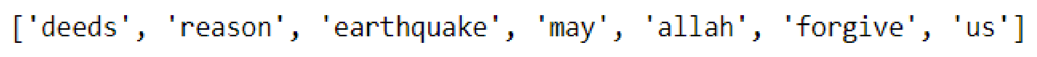


```
['deed', 'reason', 'earthquake', 'may', 'allah', 'forgive', 'u']
```

Figure 9: Lemmatizated words

f. Removal of stop words and Punctuation

There is a pre-defined stop words list in English. However, we can modify our stop words like by simply appending the words to the stop words list. From the figure 8, we can clearly see that the punctuation (#) and the words that don't add significant meaning to the text are removed from the tokenized dataset.



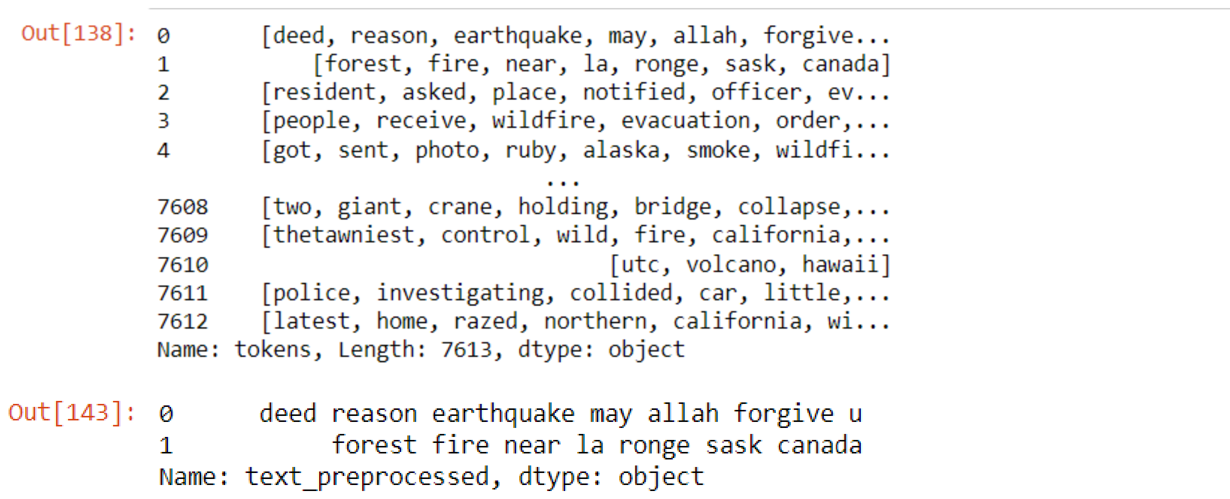
```
['deeds', 'reason', 'earthquake', 'may', 'allah', 'forgive', 'us']
```

Figure 8: Removal of stop words and punctuation

g. Vectorization of words (Bags of Words)

Bag-of-words model is mainly used as a tool of feature generation. We have converted each message which is represented by a list of tokens into a vector that a machine learning model can understood. We use SciKit Learn's CountVectorizer function which converts a collection of text documents to a matrix of token counts.

The figure 9 shows the clean twitter text after these data pre-processing steps are done.



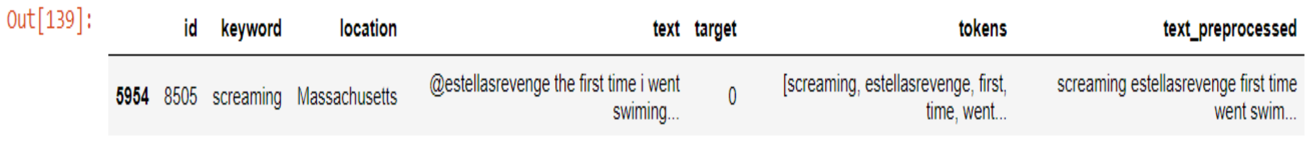
```
Out[138]: 0       [deed, reason, earthquake, may, allah, forgive...
          1           [forest, fire, near, la, ronge, sask, canada]
          2       [resident, asked, place, notified, officer, ev...
          3       [people, receive, wildfire, evacuation, order,...
          4       [got, sent, photo, ruby, alaska, smoke, wildfi...
                                        ...
          7608    [two, giant, crane, holding, bridge, collapse,...
          7609    [thetawniest, control, wild, fire, california,...
          7610                               [utc, volcano, hawaii]
          7611    [police, investigating, collided, car, little,...
          7612    [latest, home, razed, northern, california, wi...
          Name: tokens, Length: 7613, dtype: object

Out[143]: 0       deed reason earthquake may allah forgive u
          1            forest fire near la ronge sask canada
          Name: text_preprocessed, dtype: object
```

Figure 9: Cleaned dataset

2. Parameters

We use the all the parameters (Id, Keyword, Location, Text, and Target) and after the data pre-processing, we join all the parameters together as shown in figure 10.



Out[139]:

	id	keyword	location	text	target	tokens	text_preprocessed
5954	8505	screaming	Massachusetts	@estellasrevenge the first time i went swiming...	0	[screaming, estellasrevenge, first, time, went...	screaming estellasrevenge first time went swim...

Figure 10: Clean and Pre-processed dataset

We are splitting our data in to two parts 80 % for the training purpose and 20 % for the testing purpose of the model. We have also tested the training dataset which is further saved in the submission file.

Results and Discussion

To evaluate success, of implemented models we will use confusion metrics measures such as Accuracy, Precision, Recall, F1 measure. We used four different Models to get accurate data values which are Logistic regression, KNN, Multinomial Naive Bayes, Support Vector Machine. We evaluate all these four models and compare each other's to get better performance and best model which suits well on these datasets.

1. **Logistic Regression Model:**
   We use the SciKit Learn's LogisticRegression function to evaluate this model by taking random state 55 and number of iterations is 10000. We get the training accuracy 0.954 and testing accuracy 0.794 which is shown in figure 11 with confusion matrix and f1score.

```
The Train score for the Logostic Regression is:   0.9540229885057471
The Accuracy score of testing for the Logostic Regression is: 0.7944845699277742
The Binary Classification is :
              precision    recall  f1-score   support

           0       0.80      0.85      0.83       874
           1       0.78      0.72      0.75       649

    accuracy                           0.79      1523
   macro avg       0.79      0.78      0.79      1523
weighted avg       0.79      0.79      0.79      1523

The Confusion Matrix for the Logistic regression model is:
```

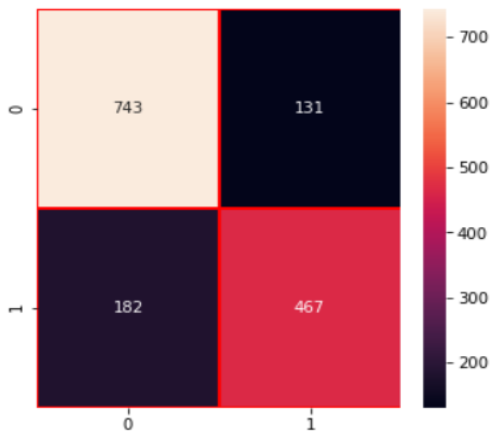


Figure 11: Confusion Matrix and Binary Classification of Logistic Regression Model

2. **K Nearest Neighbour Model**

For the KNN Model, we use the brute algorithm, nearest neighbour is 7 and weight is calculated by their distance. The training and testing accuracy score are shown in figure12 which is 0.988 and 0.730 respectively and confusion matrix and f1 score is illustrated in the same figure.

```
The Train score for the  KNN Model is:   0.9880131362889983
The Accuracy score of testing for the KNN Model is: 0.7301378857518056
The Binary Classification is :
              precision    recall  f1-score   support

           0       0.72      0.87      0.79       874
           1       0.76      0.54      0.63       649

    accuracy                           0.73      1523
   macro avg       0.74      0.71      0.71      1523
weighted avg       0.74      0.73      0.72      1523

The Confusion Matrix for the KNN model is:
```

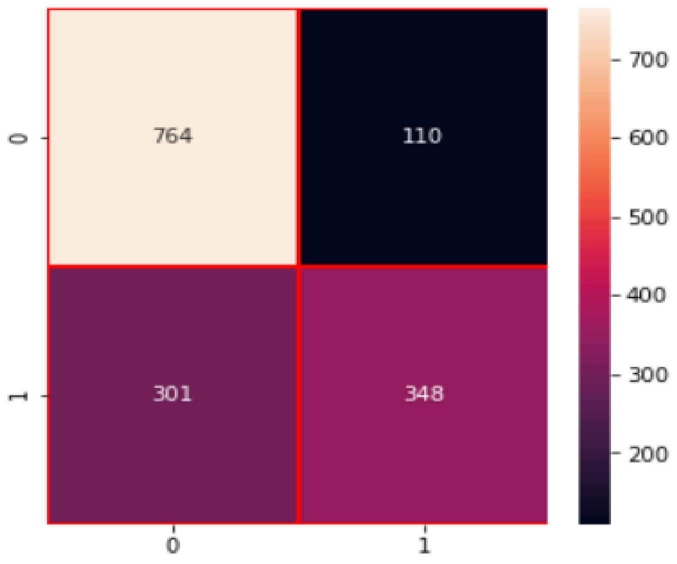


Figure 12: Confusion Matrix and Binary Classification of K Nearest Neighbour Model

3. **Multinomial Naive Bayes Model:**

The figure 13 illustrates the Training and Testing Accuracy and Binary classification and confusion matrix of the Multinomial Naïve Bayes Model. We use the learning rate 0.1 for this model. The training score is 0.937 and testing score is 0.784 which is shown in figure along the f1 score and confusion matrix.

```
The Training score for the Multinomail Naive Bayes Model is:   0.9372742200328408
The Accuracy score of testing for the Multinomail Naive Bayes Model is: 0.7846355876559422
The Binary Classification of Multinomial Naive bayes is :
              precision    recall  f1-score   support

           0       0.82      0.81      0.81       874
           1       0.74      0.76      0.75       649

    accuracy                           0.78      1523
   macro avg       0.78      0.78      0.78      1523
weighted avg       0.79      0.78      0.78      1523

The Confusion Matrix for the Multinomial Navie Bayes  model is:
```

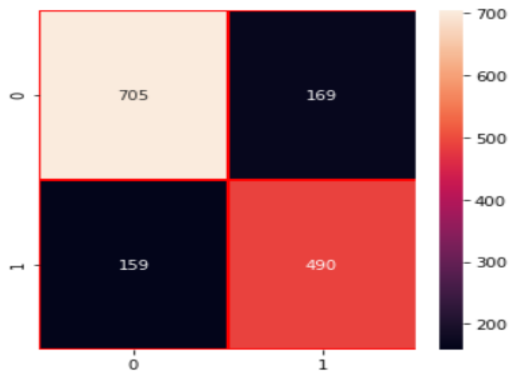


Figure 13: Confusion Matrix and Binary Classification of Multinomial Naïve Bayes

4. **Support Vector Machine Model:**

For this model, we have taken the degree 3, random state is 55 and maximum number of iterations is 10000. The figure14 shows the training and testing accuracy score 0.985 and 0.763 respectively and confusion matrix and f1 score is shown.

```
The Training score for the Support Vector Machine Model is:   0.9852216748768473
The Accuracy score of testing for the Support Vector Machine Model is: 0.7636244254760342
The Binary Classification of Support Vector Machine Model is :
              precision    recall  f1-score   support

           0       0.78      0.81      0.80       874
           1       0.73      0.70      0.72       649

    accuracy                           0.76      1523
   macro avg       0.76      0.76      0.76      1523
weighted avg       0.76      0.76      0.76      1523

The Confusion Matrix for the Support Vector Machine model is:
```

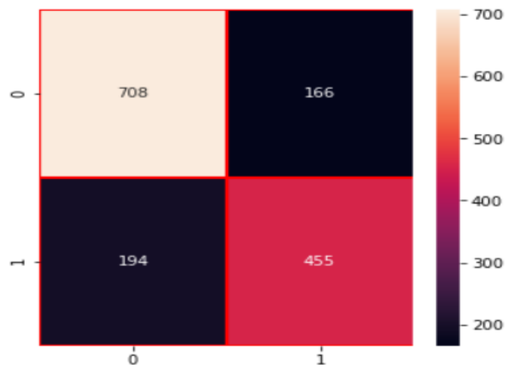


Figure 14: Confusion Matrix and Binary Classification of Support Vector Machine

CONCLUSION:

We have used the four different model to evaluate the performance, as seen in the table 3. Although the training accuracy rate of KNN and Support Vector Machine have higher than the Logistic Regression Model but both of their testing accuracy score and f1score is less than the Logistic Regression Model. We conclude that Logistic Regression model performs better than all these models with highest test accuracy rate of 0.79 and KNN has the least performance among these models with accuracy rate of 0.73. From the table3, we can clearly see that F1 score for the non-disaster tweets is higher for the Logistic Regression Model which is 0.83 and KNN has the least F1 score which is 0.79 for the target 0.

Model	Test Accuracy Score	Train Accuracy Score	Target	Recall	Precision	F1 Score
Logistic Regression	**0.794**	0.954	0	0.85	0.80	**0.83**
			1	0.72	0.78	0.75
KNN	0.730	**0.988**	0	0.87	0.72	0.79
			1	0.54	0.76	0.63
Multi Nominal Naive Bayes	0.784	0.937	0	0.81	0.82	0.81
			1	0.76	0.74	0.75
Support Vector Machine	0.763	0.985	0	0.81	0.78	0.80
			1	0.70	0.73	0.72

Table 3: Comparison of Models

In this project we have shown how to separate "disaster and Non-Disaster Tweets" which help us to find which tweets are true statement and which are not true statements to believe. This helps us to reduce the spread of false rumours into society and didn't make people panic in their life's.

Detailed Timeline and Roles:

Task	**Deadline**	**Lead**	**Status**
Exploratory Data analysis, Data pre-processing, Model implementation, visualization, calculation of all confusion matrices and comparison table	11/12/2021	Roshan Sah	Done
Prepare report	12/06/2021	Roshan & AthriNandan Goud Enuganti	Done