



INSTITUTE FOR ADVANCED COMPUTING  
AND  
SOFTWARE DEVELOPMENT AKURDI  
PUNE

Documentation On

**“Predict Tweets with Real Threats”**

PG-DBDA SEPT 2021

Submitted By:

**Group No: 05**

**AKHIL KAMBLE 219305**

**RUSHIKESH PAWAR 219339**

**Mr. Prashant Karhale**  
**Centre Coordinator**

**Mr. Akshay Tilekar**  
**Project Guide**

# Contents

<b>1. INTRODUCTION .....</b>	<b>4</b>
<b>1.1 Problem Statement .....</b>	<b>4</b>
<b>1.2 Abstract .....</b>	<b>4</b>
<b>1.3 Product Scope .....</b>	<b>4</b>
<b>1.4 Aims and Objectives .....</b>	<b>5</b>
<b>2. Overall Description .....</b>	<b>6</b>
<b>2.1 Workflow of Project: .....</b>	<b>6</b>
<b>2.2 Data Preprocessing and Cleaning .....</b>	<b>6</b>
<b>2.2.1 Data Cleaning: .....</b>	<b>6</b>
<b>2.2. Label encoding: .....</b>	<b>8</b>
<b>2.3. Exploratory Data Analysis: .....</b>	<b>9</b>
<b>2.4 Model Building .....</b>	<b>12</b>
<b>1. Train/Test split: .....</b>	<b>12</b>
<b>2. Multinomial naïve-bayes .....</b>	<b>12</b>
<b>3. BERT (Bidirectional Encoder Representations from Transformers) .....</b>	<b>13</b>
<b>3.Process Flow Diagram.....</b>	<b>16</b>
<b>4. Evaluation and Metrics .....</b>	<b>17</b>
<b>4.1 Bidirectional Encoder Representations from Transformers .....</b>	<b>17</b>
<b>4.2 Multinomial Naive Bayes .....</b>	<b>18</b>
<b>5. User Interface .....</b>	<b>19</b>
<b>6. Conclusion.....</b>	<b>21</b>
<b>7. Future Scope .....</b>	<b>22</b>
<b>8. References .....</b>	<b>23</b>

<b>Figure 1:Workflow of Project.....</b>	<b>6</b>
<b>Figure 2: Counting of each words .....</b>	<b>8</b>
<b>Figure 3: NAN values in both Test and Train set .....</b>	<b>10</b>
<b>Figure 4: Data Imbalance .....</b>	<b>10</b>
<b>Figure 5: Top 15 Keywords .....</b>	<b>11</b>
<b>Figure 6: Top Disaster and non-disaster keyword .....</b>	<b>11</b>
<b>Figure 7: Naives Bayes Classifier Diagram .....</b>	<b>13</b>
<b>Figure 8: Bert simplied Diagram.....</b>	<b>14</b>
<b>Figure 9: Bert Encoder diagram .....</b>	<b>14</b>
<b>Figure 10:Process Flow Diagram .....</b>	<b>16</b>
<b>Figure 11: Classification Report BERT .....</b>	<b>17</b>
<b>Figure 12: Confusion Matrix BERT .....</b>	<b>17</b>
<b>Figure 13:Classification Report Multinomial Naive bayes.....</b>	<b>18</b>
<b>Figure 14: The web application where the user can enter his/her Disaster tweets and press the “Predict ” Button .....</b>	<b>19</b>
<b>Figure 15:This gives us the prediction about the true disaster .....</b>	<b>20</b>
<b>Figure 16: this gives us the prediction about the false disaster .....</b>	<b>20</b>

# **1. INTRODUCTION**

## **1.1 Problem Statement**

**The main aim of our project is to distinguish if a tweet talks about a real disaster or not. This is for a competition hosted by Kaggle [5] and the dataset provided consists of a training set of 10,000 hand classified tweets on which we built our models.**

## **1.2 Abstract**

**With today's technology, each person's online footprint opens the door for a large treasure trove of information that can be used for many purposes that varies from analysing market trends to understanding the general emotion of a group of people. Twitter data is especially very useful for a variety of purposes when it comes to the latter use case, mainly because there are more than 6000 new tweets every second. With the advancement of technology and Natural Language Processing methodologies, the process of text and sentiment analysis has become much easier than a few years earlier. If in case, a person tweets a message which was about an emergency or an impending disaster and this was recognized immediately by our NLP models, we would be able to react quicker than normal which would help save lives. This is pretty much the crux of our project.**

## **1.3 Product Scope**

**The main use of this classification models is to check the quality of the question posted by the user in our web interface. The user will input the question's title and body and press the Predict Question Button after this the model will predict the quality of the question and send that quality back to the user.**

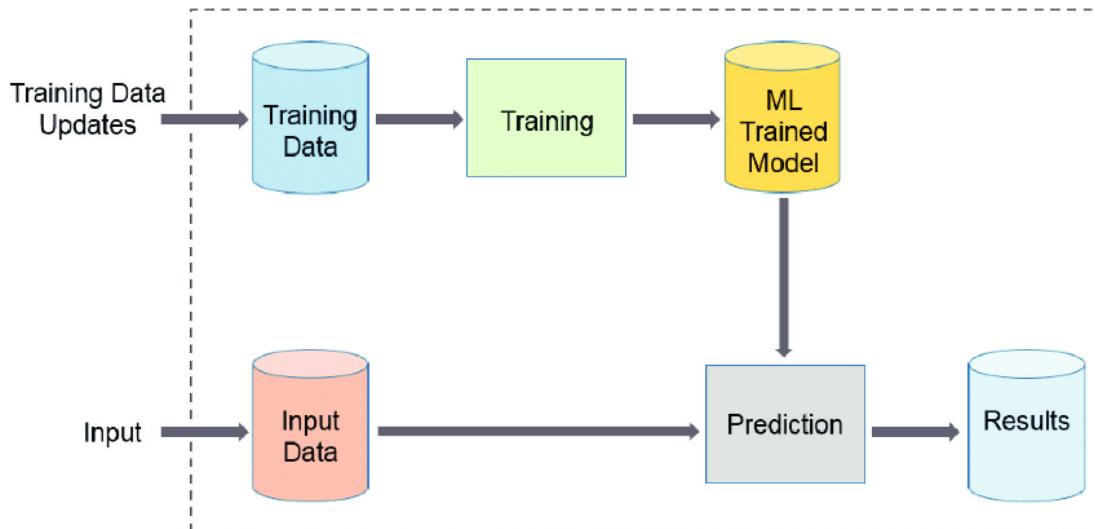
## **1.4 Aims and Objectives**

**The main aim of our project is to distinguish if a tweet talks about a real disaster or not. This is for a competition hosted by Kaggle and the dataset provided consists of a training set of 10,000 hand classified tweets on which we built our models. For the purpose of identifying if a tweet is pertaining to a disaster or not, we tried out a variety of different models like bag of words, count vectorizer followed by BERT on our BERT model as well as Multinomial Naive Bayes. Out of all these, we found that the BERT model with a Dense Classifier worked best for this dataset and gave us an f1 score of 0.98.**

## 2. Overall Description

### 2.1 Workflow of Project:

The diagram below shows the workflow of this project.



*Figure  
1: Workflow  
of Project*

### 2.2 Data Preprocessing and Cleaning

#### 2.2.1 Data Cleaning:

Before we start, we pre-processed the data to get it all in a consistent format. We cleaned, tokenized and converted our data into a matrix. Some of the basic text pre-processing techniques includes: Make text all lowercase or uppercase so that the algorithm does not treat the same words in different cases as different. Removing Noise i.e., everything that isn't in a standard number or letter i.e. Punctuation, Numerical values, common nonsensical text (like /n)

- **Stop word Removal**

Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words.

- **Stemming**

**Stemming is the process of reducing inflected (or sometimes derived) words to their stem, base or root form — generally a written word form. Example if we were to stem the following words: “Stems”, “Stemming”, “Stemmed” and “Stigmatization”, the result would be a single word “stem”.**

- **Lemmatization**

**A slight variant of stemming is lemmatization. The major difference between these is that stemming can often create non-existent words, whereas lemmas are actual words. So, your root stem, meaning the word you end up with, is not something you can just look up in a dictionary, but you can look up a lemma. Examples of Lemmatization are that “run” is a base form for words like “running” or “ran” or that the word “better” and “good” are in the same lemma, so they are considered the same.**

- **Tokenization**

**Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e., words that we actually want. Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings. We used various tokenizer and word embeddings such as count vectors, TF-IDF vectorization, Continuous Bag of words, Fast text.**

**The next step is to normalize the count matrix using TF-IDF representation. The main reason behind normalizing frequencies instead of using the raw ones is to decrease the effect of tokens that occur several times in the text, and there’s not as informative as those presented a few times. For instance, the word “document” occurs a thousand times in a given corpus, and “awesome” happens two times. TF-IDF works in this situation, like pre-processing data to change raw feature vectors into a representation that is more suitable for machine learning algorithms.**

## 2.2. Label encoding:

To make the data understandable or in human readable form, the training data is often labelled in words. Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated.

- Transforming tokens to a vector

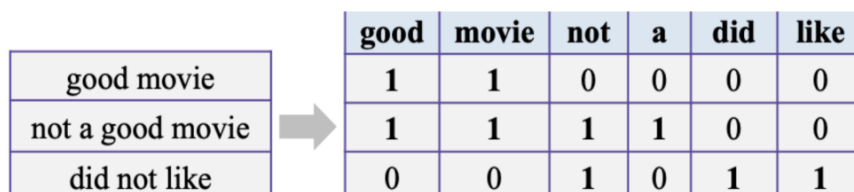
After the initial pre-processing phase, we need to transform text into a meaningful vector (or array) of numbers. This can be done by a number of techniques:

### Bag of Words

The bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

Why is it called a “bag” of words? That is because any information about the order or structure of words in the document is discarded and the model is only concerned with whether the known words occur in the document, not where they occur in the document.



	good	movie	not	a	did	like
good movie	1	1	0	0	0	0
not a good movie	1	1	1	1	0	0
did not like	0	0	1	0	1	1

*Figure 2: Counting of each words*

We can do this using scikit-learn's CountVectorizer, where every row will represent a different tweet and every column will represent a different word.

- Bag of Words - Countvectorizer Features  
Countvectorizer converts a collection of text documents to a matrix of token counts. It is important to note here that CountVectorizer comes with a lot of options to automatically do pre-processing, tokenization, and stop word



removal. However, all the processes were done manually to just get a better understanding.

- **TF-IDF Features**

A problem with the Bag of Words approach is that highly frequent words start to dominate in the document (e.g., larger score), but may not contain as much “informational content”. Also, it will give more weight to longer documents than shorter documents. One approach is to rescale the frequency of words by how often they appear in all documents so that the scores for frequent words like “the” that are also frequent across all documents are penalized. This approach to scoring is called Term Frequency-Inverse Document Frequency, or TF-IDF for short, where:

*Term Frequency:* is a scoring of the frequency of the word in the current document.

$TF = (\text{Number of times term } t \text{ appears in a document}) / (\text{Number of terms in the document})$

*Inverse Document Frequency:* is a scoring of how rare the word is across Documents.

$IDF = 1 + \log(N/n),$

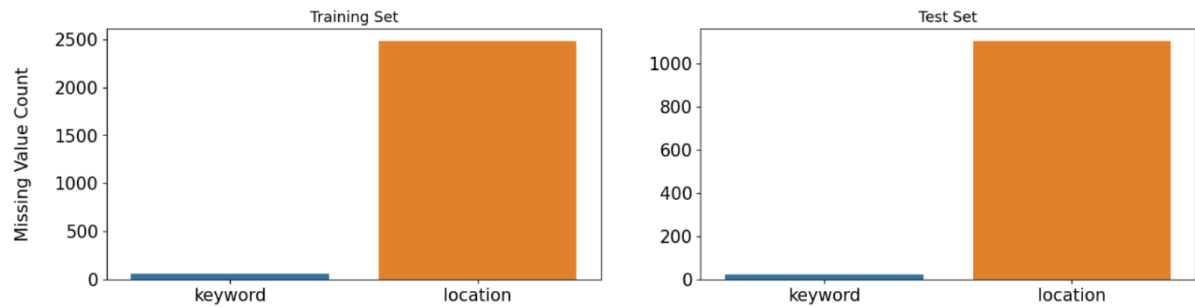
where  $N$  is the number of documents and  $n$  is the number of documents a term  $t$  has appeared in.

### **2.3. Exploratory Data Analysis:**

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Following are some plots we used to extract some useful information

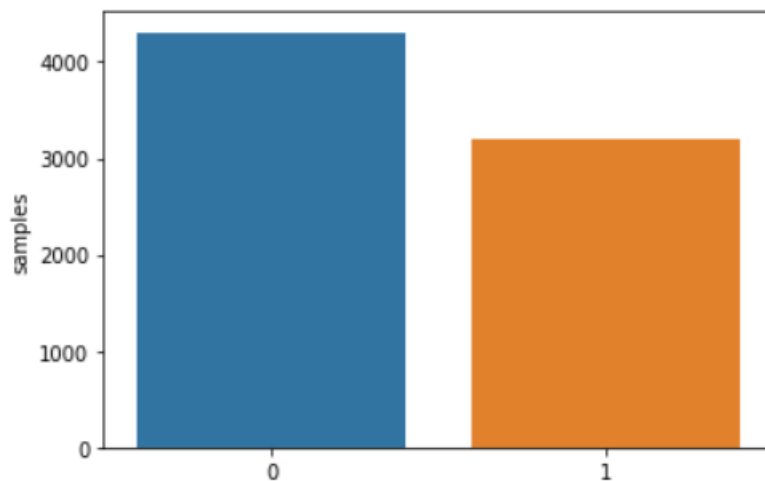
## A) Keyword and Location



*Figure 3: NAN values in both Test and Train set*

**Exp:** This Plot shows the ammount of missing values from Keyword and Location Columns in the dataset

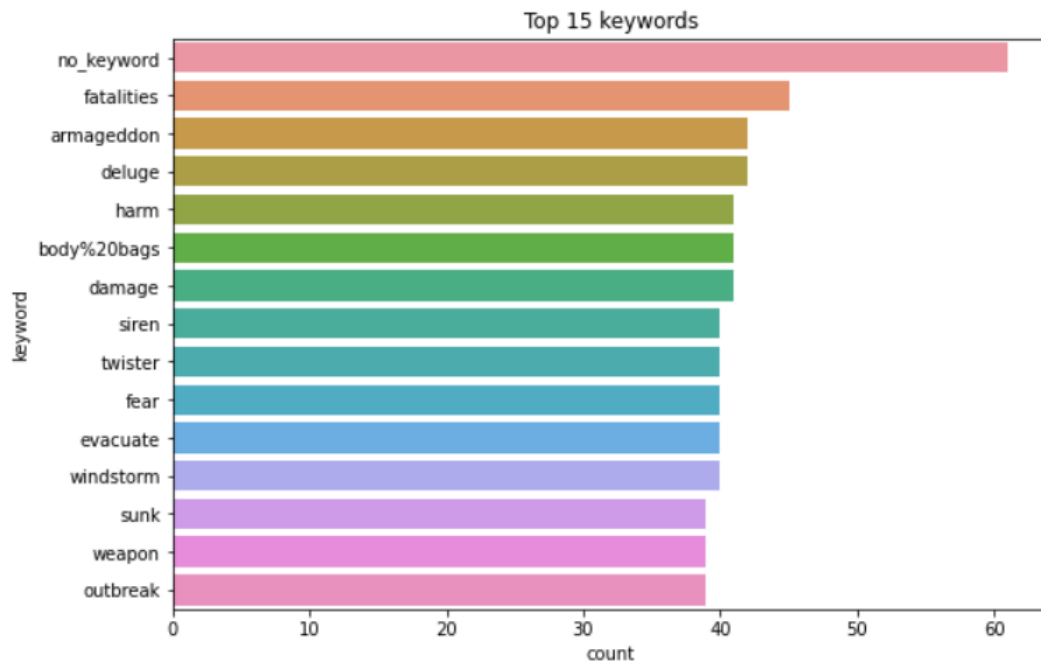
## B) Data Imbalance



*Figure 4: Data Imbalance*

**Exp:** To see if the data is balanced or unbalanced

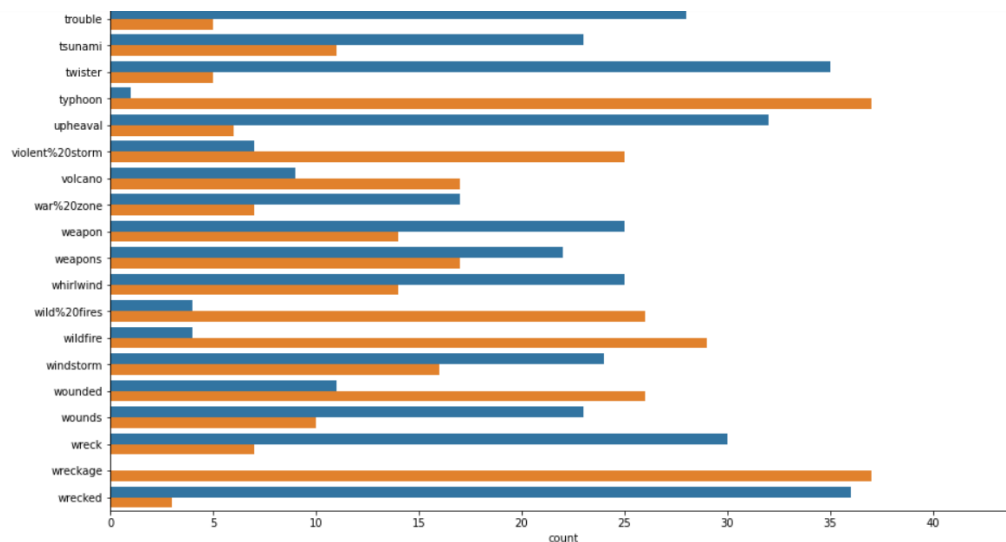
### C) Top 15 Keywords



*Figure 5: Top 15 Keywords*

**Exp:** This shows top 15 most occuring words and their count from Keyword column

### D) Top Disaster and Non Disaster Keyword



*Figure 6: Top Disaster and non-disaster keyword*

**Exp:** This chart shows how many times a word has been used in disaster and non disaster tweet

## **2.4 Model Building**

### **1. Train/Test split:**

**One important aspect of all machine learning models is to determine their accuracy. Now, in order to determine their accuracy, one can train the model using the given dataset and then predict the response values for the same dataset using that model and hence, find the accuracy of the model. A better option is to split our data into two parts: first one for training our machine learning model, and second one for testing our model.**

- **Split the dataset into two pieces: a training set and a testing set.**
- **Train the model on the training set.**
- **Test the model on the testing set, and evaluate how well our model did.**

#### **Advantages of train/test split:**

- **Model can be trained and tested on different data than the one used for training.**
- **Response values are known for the test dataset, hence predictions can be evaluated**
- **Testing accuracy is a better estimate than training accuracy of out-of-sample performance.**

**Machine learning consists of algorithms that can automate analytical model building. Using algorithms that iteratively learn from data, machine learning models facilitate computers to find hidden insights from Big Data without being explicitly programmed where to look.**

**We have used the following three algorithms to build predictive model.**

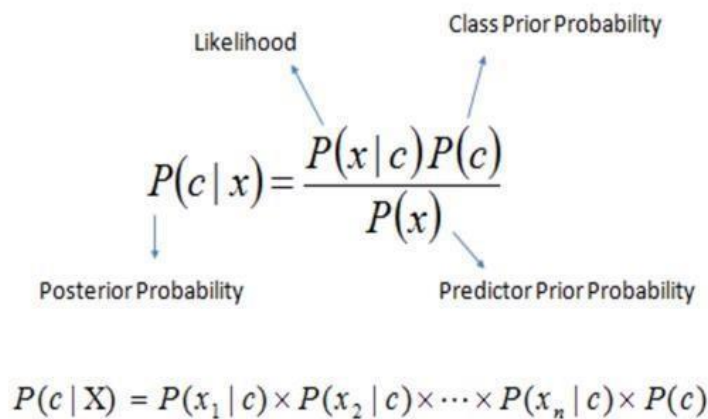
### **2. Multinomial naïve-bayes**

**To begin our analysis, we considered Naive Bayes as an initial model, a popular classifier for unstructured text with proven ability in the field.**

**Naive Bayes performs well with a small amount of data and uses prior knowledge to calculate a posterior probability, represented by a probability**

distribution which reflects the likelihood that a specific instance belongs to a particular class. With our data prepared we can instantiate a Naive Bayes classifier by invoking an SKLearn class, initially maintaining default parameters to establish a baseline performance. We will use this baseline performance as a means of deciding whether to further explore the parameter space of Naive Bayes or consider a different model for our problem.

## Naive Bayes Classifier



The diagram shows the Naive Bayes equation:  $P(c | x) = \frac{P(x | c)P(c)}{P(x)}$ . Arrows point from labels to parts of the equation: 'Likelihood' points to  $P(x | c)$ , 'Class Prior Probability' points to  $P(c)$ , 'Posterior Probability' points to  $P(c | x)$ , and 'Predictor Prior Probability' points to  $P(x)$ . Below the equation is the expanded formula:  $P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$ .

Figure 7: Naives Bayes Classifier Diagram

### 3. BERT (Bidirectional Encoder Representations from Transformers)

This model utilizes the implementation of BERT at TensorFlow/models/official/nlp/bert from the TensorFlow Models repository on GitHub. We use L=12 hidden layers, a hidden size of H=768, and A=12 focus heads (Transformer blocks). On Wikipedia and Books Corpus, this model was pre-trained for English. Inputs have been "uncased" indicating that before tokenization into word bits, the text has been lower-cased, and all accent marks have been deleted. To download this model, the kernel needs to be activated on the Internet.

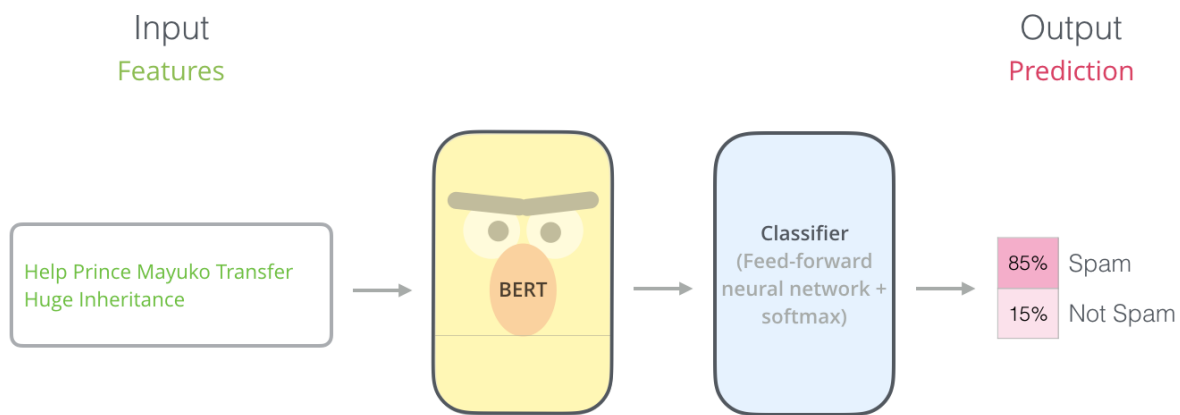


Figure 8: Bert simplified Diagram

## 1. BI-DIRECTIONAL

Bi-directionality means it can see the inputs from both directions and even from between, which was not possible in previous models as they were uni-directional(from left to right/right to left), due to which the model was unable to see the entire sentences which lead to biased representation of word(encodings).

## 2. ENCODER

The main purpose of the BERT is to generate encoding/ embeddings in a way such that they are not biased and later use them in tasks of the desired type, due to these type of embedding model learns the general patters of how human talks and generate responses/output based on that which are not biased.

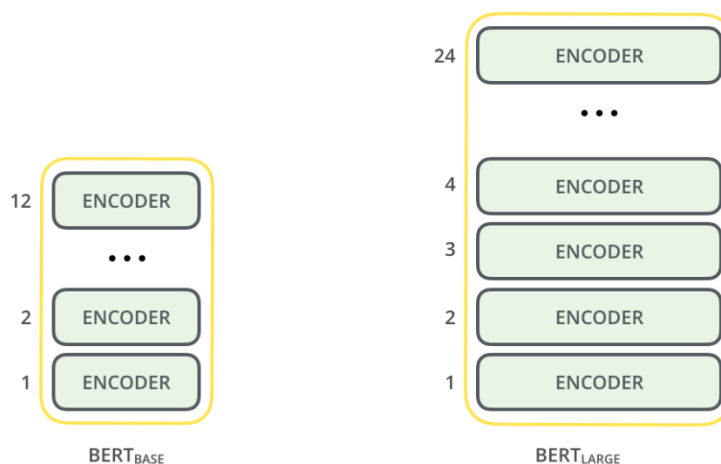


Figure 9: Bert Encoder diagram

To do this the BERT model consists of multiple encoder layers whose work is to do pre-processing and then encode the inputs in some form of embeddings which can later be used by some model

### **3. TRANSFORMER**

Bert is from the transformer family but different in the way that it only has encoder blocks and uses only attention and feed-forward layer to generate segment embeddings. Also, the hyperparameters for the attention head are different(usually 12-16 attention heads) compared to the original transformer model.

**Bert is available in 2 variations:**

**BERT\_Large (Archive SOTA results)**

**Layers – 24**

**Hidden State – 1024**

**Self-Attention Heads – 16**

**Total Parameters – 340 Million**

**BERT\_base (small model)**

**Layers – 12**

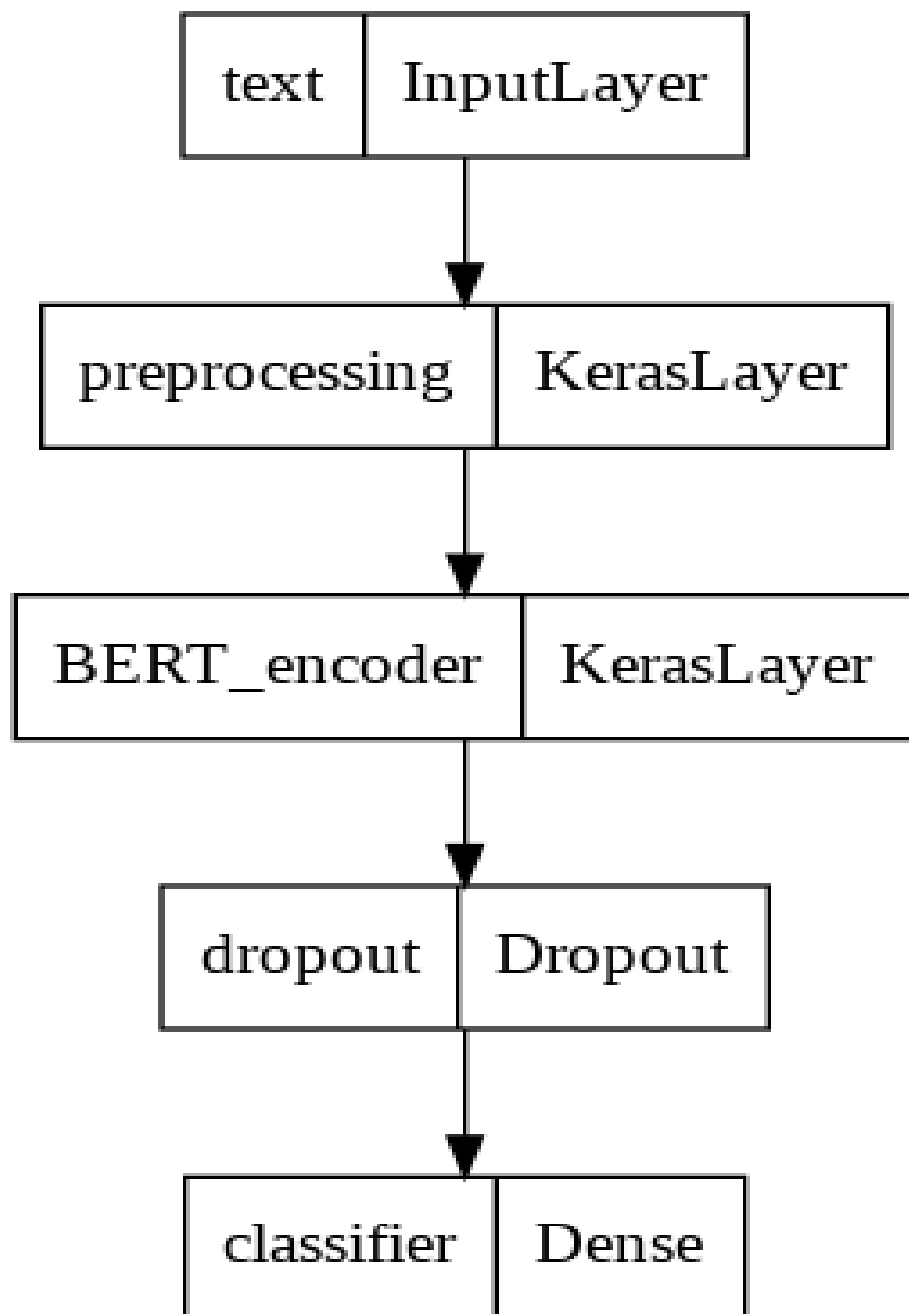
**Hidden State – 768**

**Self-Attention Heads – 12**

**Total Parameters – 110 Million**

**For most of the use cases, BERT\_base is sufficient and provides good results on finetuning.**

### 3.Process Flow Diagram



*Figure 10:Process Flow Diagram*



## 4. Evaluation and Metrics

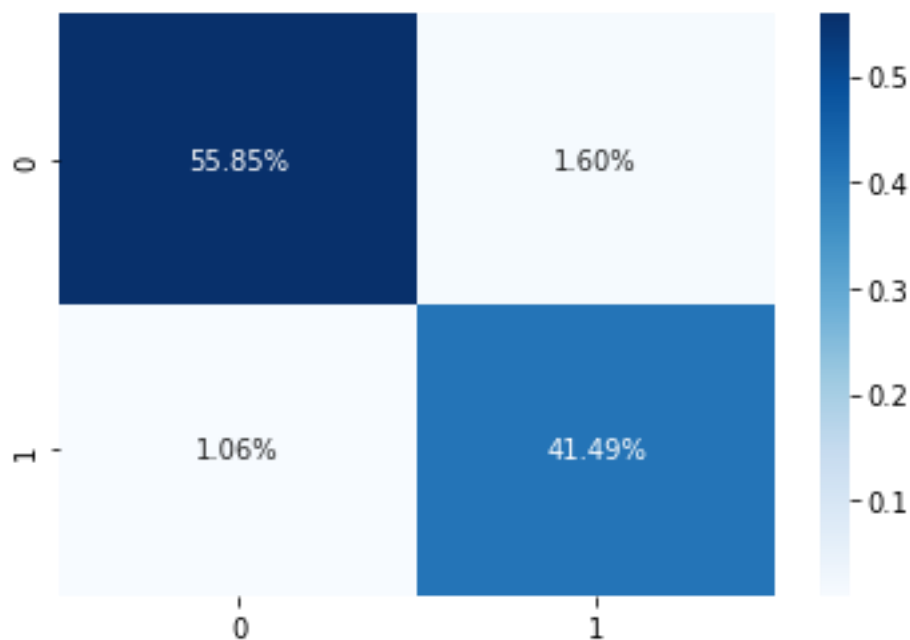
### 4.1 Bidirectional Encoder Representations from Transformers

#### A) Classification Report

	precision	recall	f1-score	support
0	0.98	0.99	0.98	216
1	0.98	0.97	0.98	160
accuracy			0.98	376
macro avg	0.98	0.98	0.98	376
weighted avg	0.98	0.98	0.98	376

*Figure 11: Classification Report BERT*

#### B) Confusion Matrix



*Figure 12: Confusion Matrix BERT*

## 4.2 Multinomial Naive Bayes

### A) Classification Report

	precision	recall	f1-score	support
0	0.77	0.95	0.85	216
1	0.90	0.62	0.74	160
accuracy			0.81	376
macro avg	0.84	0.79	0.80	376
weighted avg	0.83	0.81	0.80	376

Figure 13: Classification Report Multinomial Naive bayes

## 5. User Interface

After Training the models and finding out the best model to be BERT we can go ahead with building the user interface for our models. This will give us a clean and simple way of accessing our models and make the predictions on our questions.

For building the user interface we adopted the Flask Web Framework which is very easy to use and robust.

Here are a few screenshots of the web app delivering the “Predict Tweets with Real Threats”

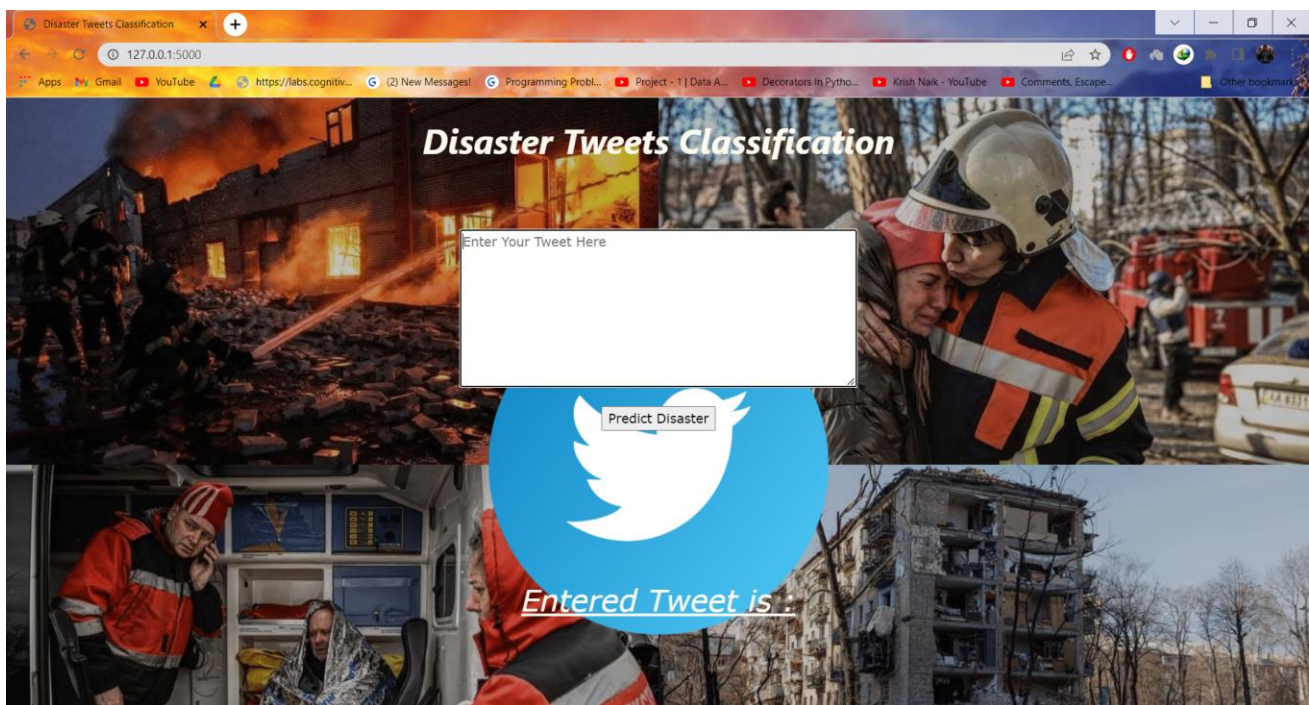


Figure 14: The web application where the user can enter his/her Disaster tweets and press the “Predict ” Button



Figure 15: This gives us the prediction about the true disaster



Figure 16: this gives us the prediction about the false disaster

## **6. Conclusion**

- 1) In this project we have built a system for classifying a tweet into real disaster tweet or not.**
- 2) For Model building we gathered the data from Kaggle.com**
- 3) We cleaned the data of any special characters. Performed Label Encoding on the response variable**
- 4) We did Exploratory Data Analysis to get inputs regarding the dataset**
- 5) Built a BERT model and Multinomial Naive Bayes on the cleaned dataset. Found out BERT to be the best model.**

## **7. Future Scope**

- 1) We can build a web app to input single tweets and get the output in an instance with the output probability.**
- 2) We can increase the accuracy even more by using other BERT models depending on the need and usage.**

## 8. References

- BERT tutorial on Tensorflow documentation  
[https://www.tensorflow.org/tutorials/keras/text\\_classification](https://www.tensorflow.org/tutorials/keras/text_classification)
- Jay Alammar blog on BERT  
<https://jalammar.github.io/illustrated-bert/>
- Kaggle Competition Dataset  
<https://www.kaggle.com/competitions/nlp-getting-started/data?select=test.csv>