# UG/B. TECH PROJECT REPORT

on

# Sentiment Analysis - Disaster Tweets Classification

By

**Mr Piyush Pransukhka**
20124035
Mathematics and Computing
Indian Institute of Technology (BHU)
Varanasi - 221 005

**Mr Shrey Gupta**
20124045
Mathematics and Computing
Indian Institute of Technology (BHU)
Varanasi - 221 005

# Introduction and Motivation

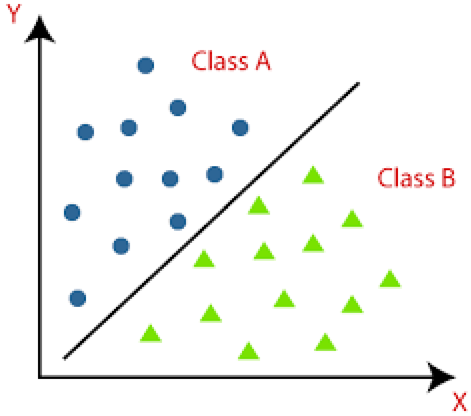
Sentiment analysis is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text.

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. But, it’s not always clear whether a person’s words are actually announcing a disaster.

We will build an ML model to predict which Tweets are about real disasters and which ones aren’t. It can be easily seen that this is a Binary Classification task along with some concepts of NLP (Natural Language Processing).

# Binary Classification

Binary classification is the task of classifying the elements of a set into two groups on the basis of a classification rule. Logistic Regression is used for binary classification. Logistic Regression is a statistical model to predict the probability of an event occurring.



$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b), \text{ where } \sigma(z^{(i)}) = \frac{1}{1 + e^{-z^{(i)}}}$$

$\hat{y}^{(i)}$ represents the probability that a prediction is positive, i.e. P(Y=1| X) on the i$^{th}$ training example.

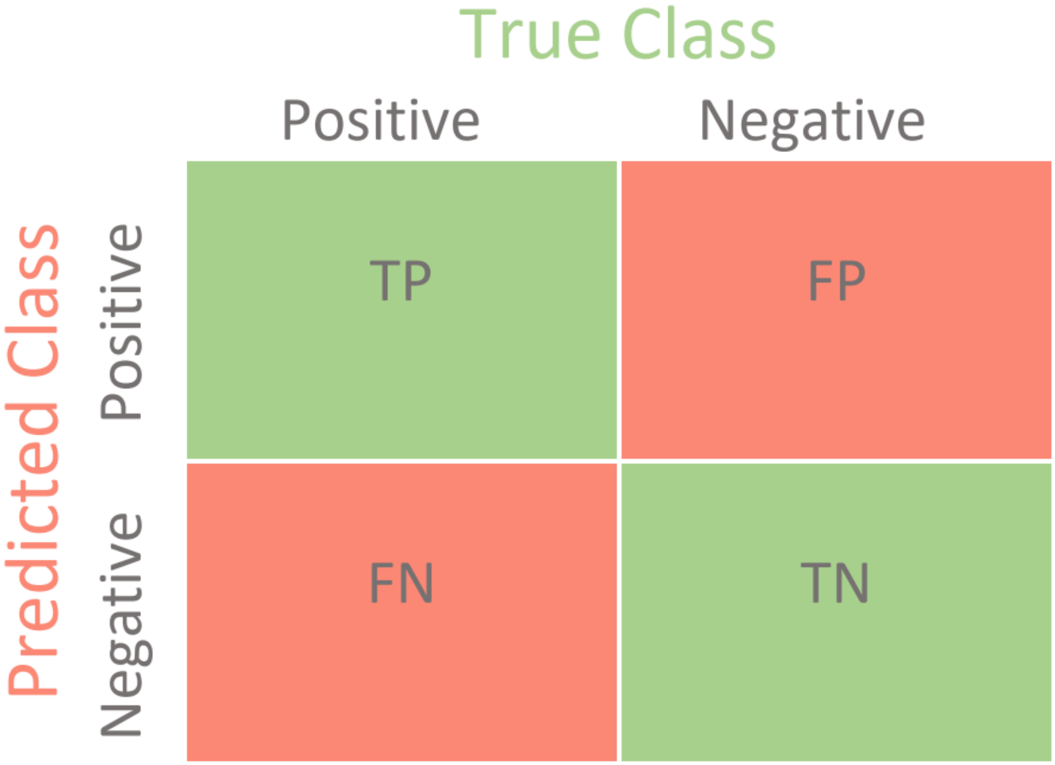
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[( y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})]$$

where, the cost function *J(w,b)* is defined as the average of the loss function over all *m* training examples.
The aim of the logistic regression is to reduce the cost function to get better results.

# Evaluation of Binary Classification

### Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm.



| | | True Class | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted Class | Positive | TP | FP |
| | Negative | FN | TN |

### Accuracy

Accuracy is a metric for classification models that measures the number of predictions that are correct as a percentage of the total number of predictions that are made.

$$Accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ total\ predictions}$$

### Precision

Precision attempts to answer the following question: What proportion of positive identifications was actually correct?

Precision is defined as follows:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

### Recall

Recall attempts to answer the following question: What proportion of actual positives was identified correctly?

Mathematically, recall is defined as follows:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

### F1 Score

There is a tradeoff between precision and recall, so to overcome this issue we introduce a new metric F1 score which takes into account both precision and recall.

$$\begin{aligned}\text{F1 Score} &= \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \\ &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\end{aligned}$$

# Sequential Models

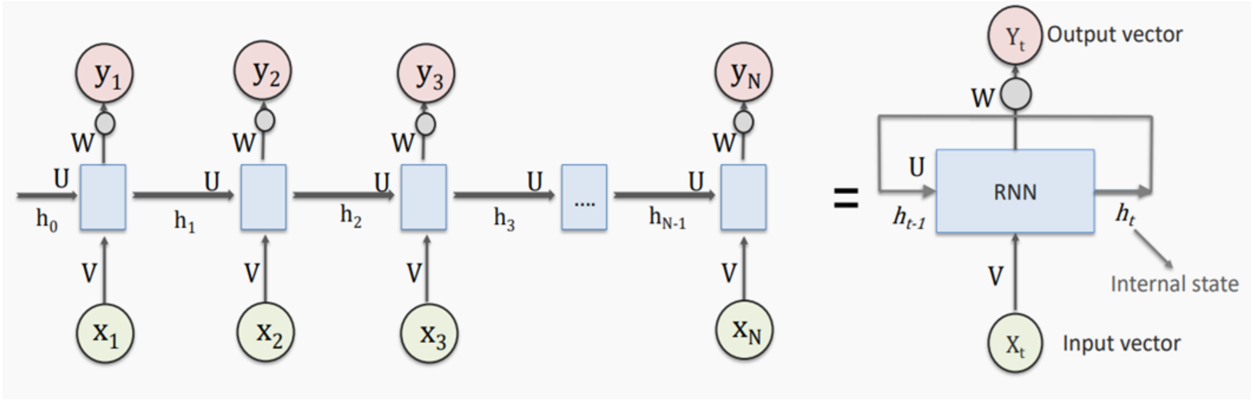
Sequential models are machine learning models that input or output sequences of data. Sequential data includes text streams, audio clips, video clips, time-series data and etc.

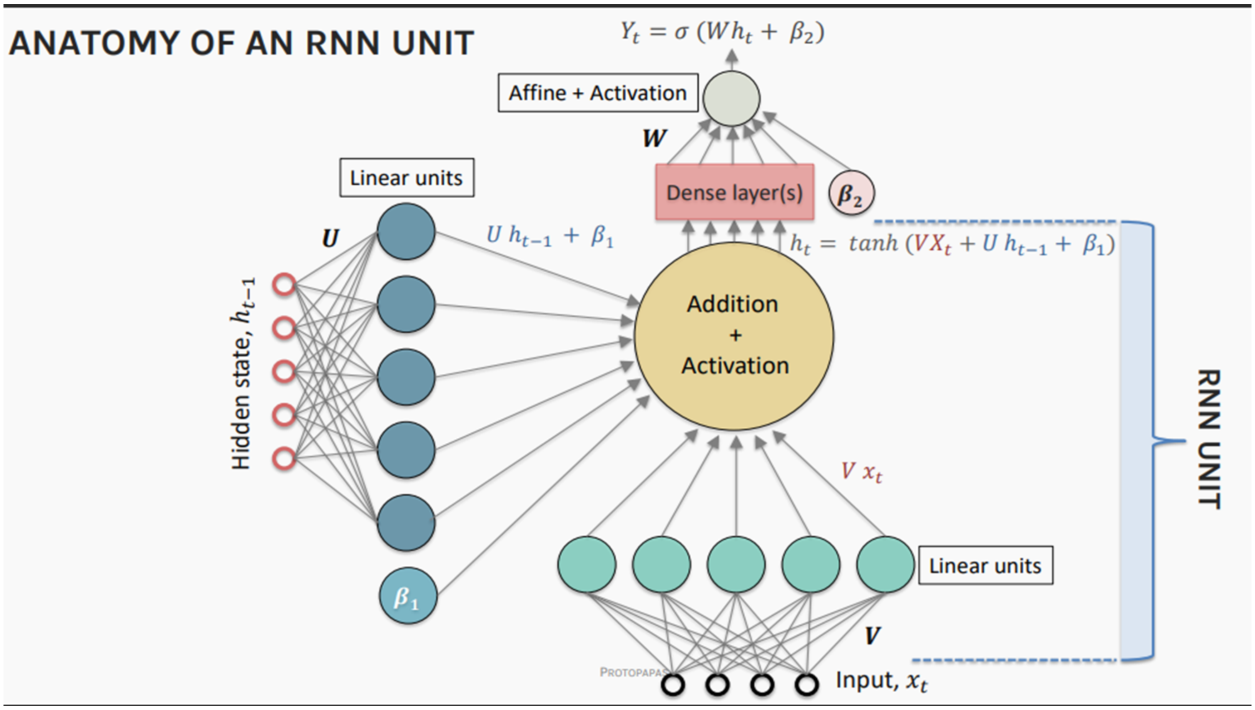
We can implement such sequential models using:-

- RNNs
- GRUs
- LSTMs

# Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a type of artificial neural network that uses sequential data or time series data.
RNNs are governed by a recurrence relation applied at every time step for a given sequence.



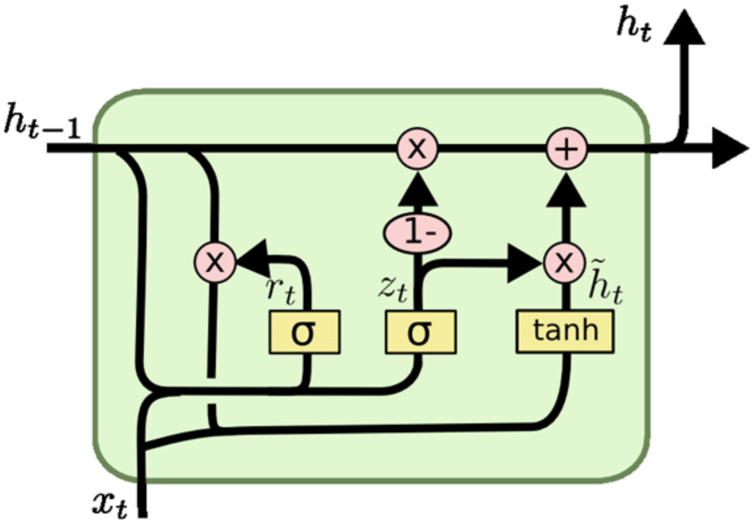


**Drawbacks of RNNs:**

- Training RNNs
- The vanishing or exploding gradient problem
- RNNs cannot be stacked up
- Slow and Complex training procedures
- Difficult to process longer sequences

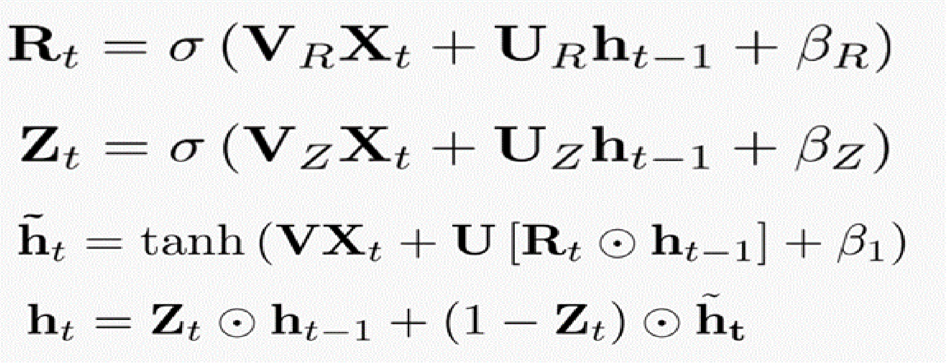
# Gated Recurrent Units (GRUs)

A gated neural network uses processes known called update gate and reset gate. This allows the neural network to carry information forward across multiple units by storing values in memory. When a critical point is reached, the stored values are used to update the current state. The memory is carried using the hidden state ($h_t$).



A GRU has two gates:

1) Update Gate - This gate updates the weight of the units and thus resolving the issue of vanishing gradients.
2) Reset Gate - This gate forgets the information from the previous cells.



$$\mathbf{R}_t = \sigma\left(\mathbf{V}_R \mathbf{X}_t + \mathbf{U}_R \mathbf{h}_{t-1} + \beta_R\right)$$

$$\mathbf{Z}_t = \sigma\left(\mathbf{V}_Z \mathbf{X}_t + \mathbf{U}_Z \mathbf{h}_{t-1} + \beta_Z\right)$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{V}\mathbf{X}_t + \mathbf{U}\left[\mathbf{R}_t \odot \mathbf{h}_{t-1}\right] + \beta_1\right)$$

$$\mathbf{h}_t = \mathbf{Z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{h}}_{\mathbf{t}}$$

### Drawbacks of GRUs

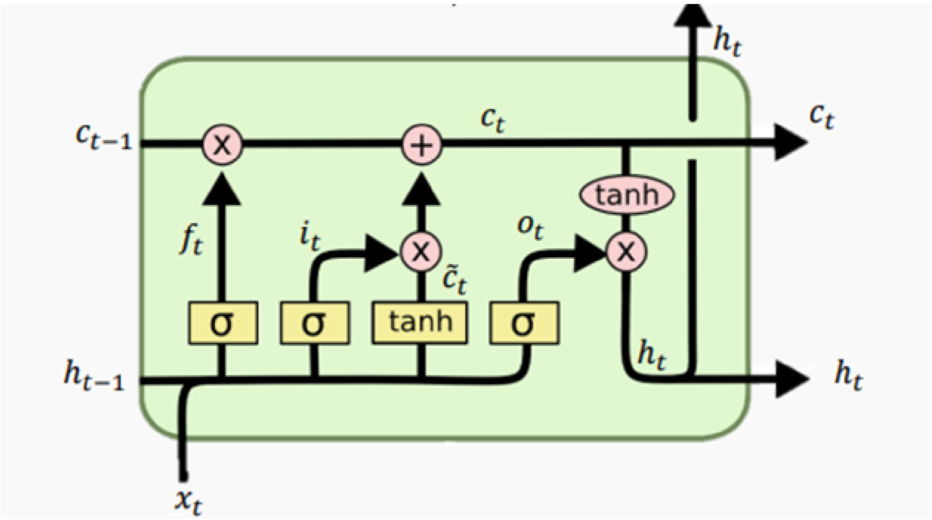
- The same hidden state is used for both memory and output.
- For longer sequences, two gates may be insufficient.

# Long Short-Term Memory (LSTMs)

LSTM is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. It uses three gates - input gate, output gate and forget gate. Here a new state called cell state ($c_t$) is introduced to create a difference between output and memory.

The LSTM has three gates:

1) Input Gate: It extracts only useful information from the current state.
2) Forget Gate: It forgets some information from the previous states.
3) Output Gate: It is used to create a difference between memory and output.



$$\mathbf{f}_t = \sigma\left(\mathbf{V}_f\mathbf{X}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \beta_f\right)$$

$$\mathbf{i}_t = \sigma\left(\mathbf{V}_i\mathbf{X}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \beta_i\right)$$

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{V}\mathbf{X}_t + \mathbf{U}\mathbf{h}_{t-1} + \beta_1\right)$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{o}_t = \sigma\left(\mathbf{V}_o\mathbf{X}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \beta_o\right)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)$$

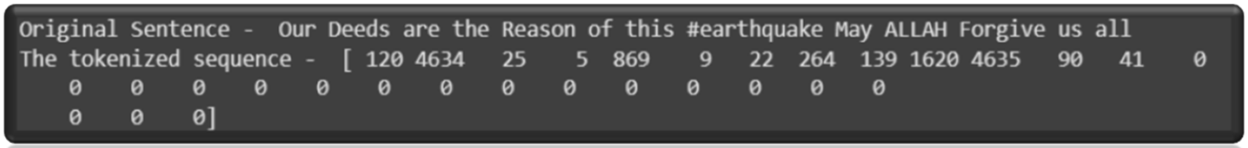
# TOKENIZER

Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or subwords.

Given a sentence or paragraph, a space tokenizer tokenizes into words by splitting the input whenever a white space is encountered.

The tokenizer in Tensorflow tokenizes the sentence as well as assigns a unique number to each token.

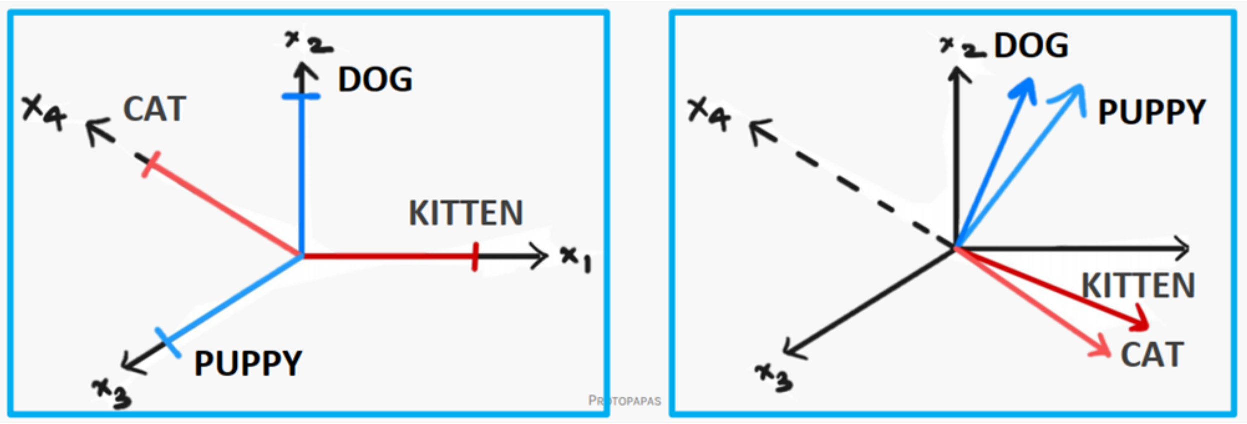
An example of a sentence being tokenized and padded with zeros is:

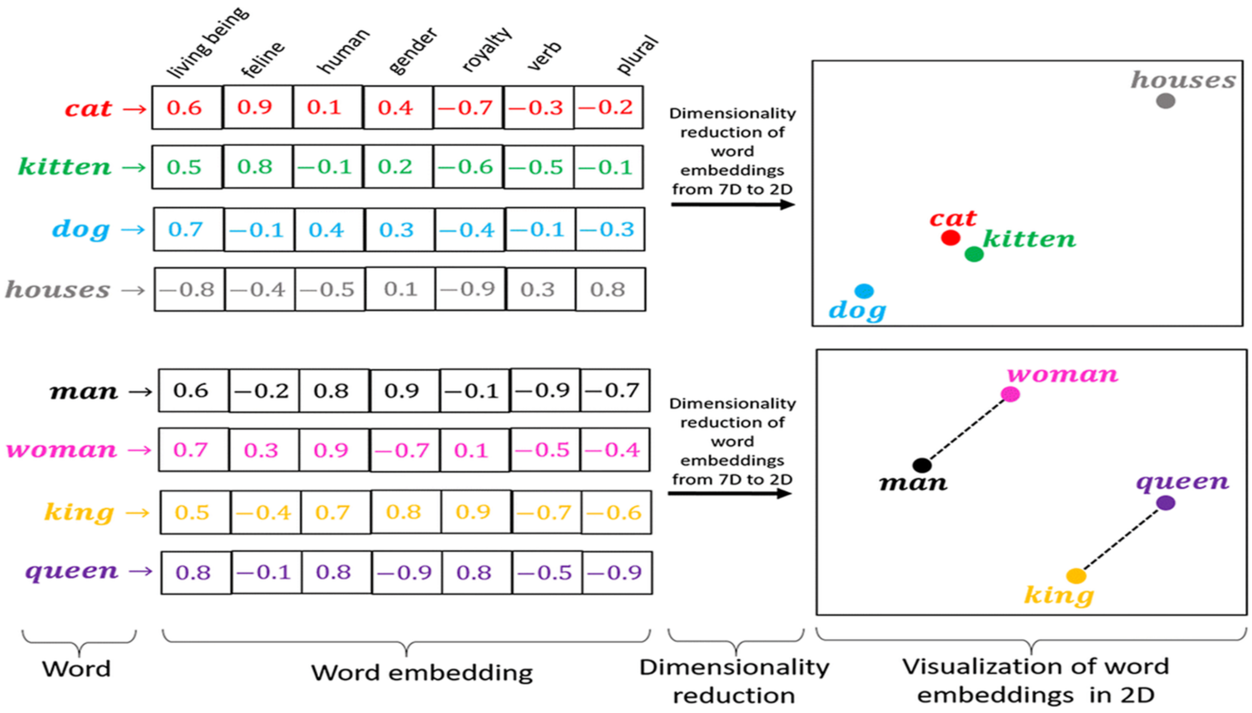


```
Original Sentence -  Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all
The tokenized sequence -  [ 120 4634   25    5  869    9   22  264  139 1620 4635   90   41    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0]
```

# Word Embeddings

Word embedding is a representation of a word. The embedding is used in text analysis. Typically, the representation is a real-valued vector that encodes the meaning of the word in such a way that words that are closer in the vector space are expected to be similar in meaning.

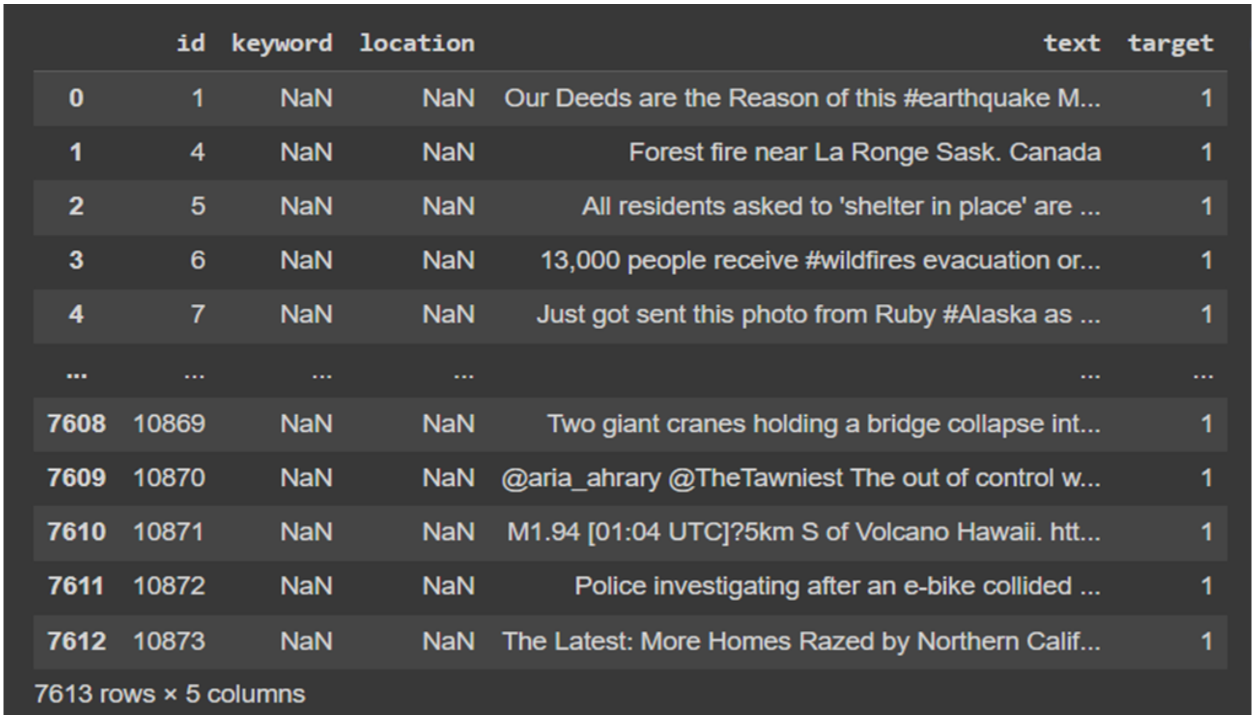




| Word | living being | feline | human | gender | royalty | verb | plural |
|---|---|---|---|---|---|---|---|
| cat → | 0.6 | 0.9 | 0.1 | 0.4 | −0.7 | −0.3 | −0.2 |
| kitten → | 0.5 | 0.8 | −0.1 | 0.2 | −0.6 | −0.5 | −0.1 |
| dog → | 0.7 | −0.1 | 0.4 | 0.3 | −0.4 | −0.1 | −0.3 |
| houses → | −0.8 | −0.4 | −0.5 | 0.1 | −0.9 | 0.3 | 0.8 |
| man → | 0.6 | −0.2 | 0.8 | 0.9 | −0.1 | −0.9 | −0.7 |
| woman → | 0.7 | 0.3 | 0.9 | −0.7 | 0.1 | −0.5 | −0.4 |
| king → | 0.5 | −0.4 | 0.7 | 0.8 | 0.9 | −0.7 | −0.6 |
| queen → | 0.8 | −0.1 | 0.8 | −0.9 | 0.8 | −0.5 | −0.9 |

Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

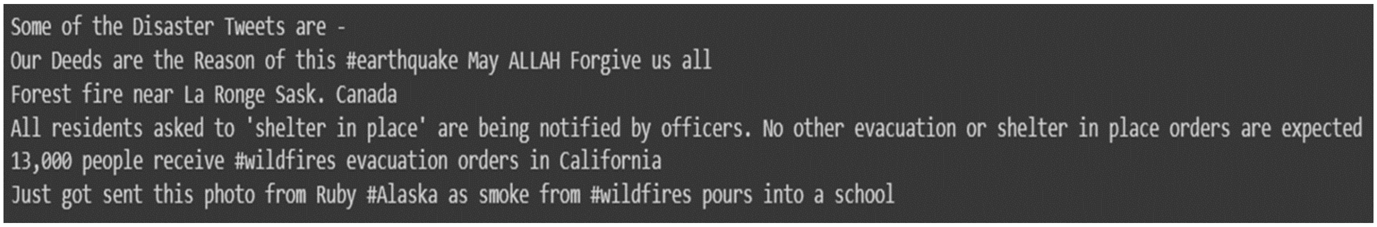
# Exploratory Data Analysis



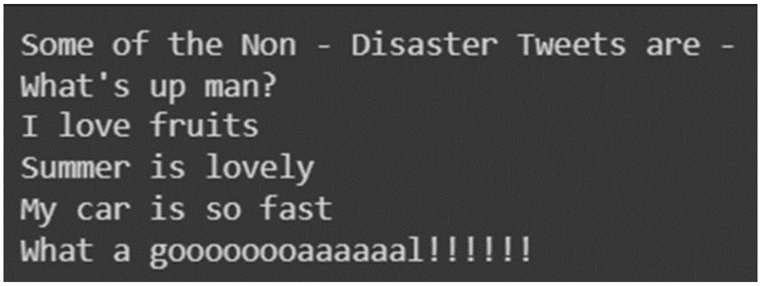
| | id | keyword | location | text | target |
|---|---|---|---|---|---|
| 0 | 1 | NaN | NaN | Our Deeds are the Reason of this #earthquake M... | 1 |
| 1 | 4 | NaN | NaN | Forest fire near La Ronge Sask. Canada | 1 |
| 2 | 5 | NaN | NaN | All residents asked to 'shelter in place' are ... | 1 |
| 3 | 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 |
| ... | ... | ... | ... | ... | ... |
| 7608 | 10869 | NaN | NaN | Two giant cranes holding a bridge collapse int... | 1 |
| 7609 | 10870 | NaN | NaN | @aria_ahrary @TheTawniest The out of control w... | 1 |
| 7610 | 10871 | NaN | NaN | M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt... | 1 |
| 7611 | 10872 | NaN | NaN | Police investigating after an e-bike collided ... | 1 |
| 7612 | 10873 | NaN | NaN | The Latest: More Homes Razed by Northern Calif... | 1 |

7613 rows × 5 columns

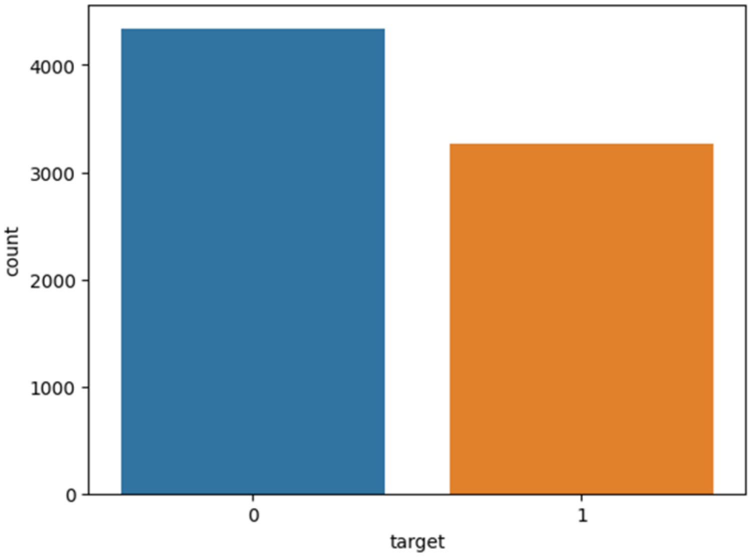
The training data consists of 7613 data points/ training examples. Each data point consists of 5 columns namely, *id, keyword, location, text and target,* out of which the first 3 columns are not of much use. The text column represents the tweet and the target value is 1 if the text is disastrous and 0 if non-disastrous.



```
Some of the Disaster Tweets are -
Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all
Forest fire near La Ronge Sask. Canada
All residents asked to 'shelter in place' are being notified by officers. No other evacuation or shelter in place orders are expected
13,000 people receive #wildfires evacuation orders in California
Just got sent this photo from Ruby #Alaska as smoke from #wildfires pours into a school
```



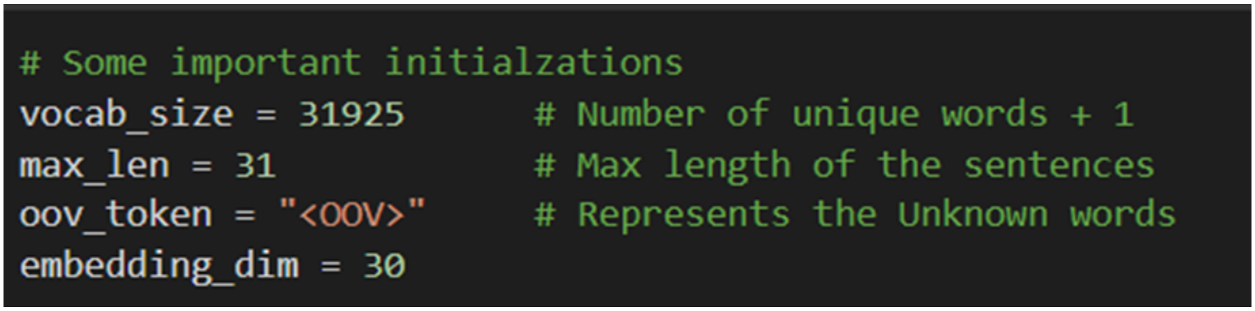
```
Some of the Non - Disaster Tweets are -
What's up man?
I love fruits
Summer is lovely
My car is so fast
What a goooooooaaaaaal!!!!!!
```



The training data consists of around 4000 non-disastrous tweets and around 3000 disastrous tweets. Hence the data is quite equally distributed.

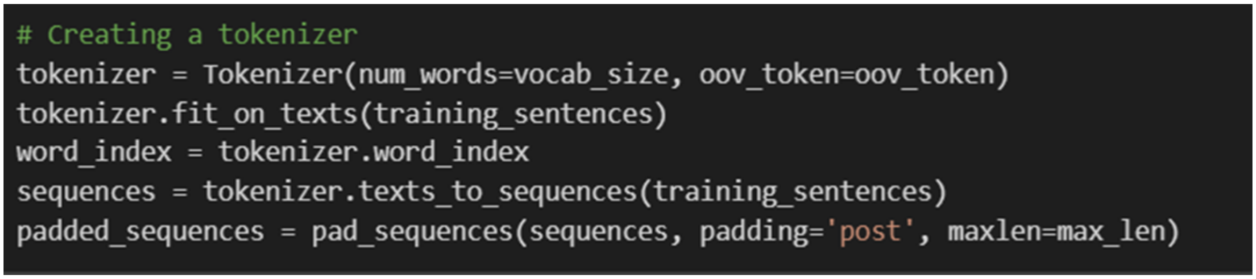
# Creating Training Data

We first started with some basic initializations as follows:



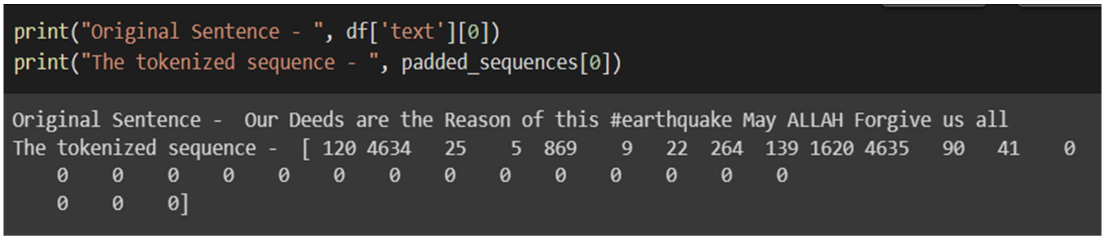
```
# Some important initialzations
vocab_size = 31925      # Number of unique words + 1
max_len = 31            # Max length of the sentences
oov_token = "<OOV>"     # Represents the Unknown words
embedding_dim = 30
```

Then we created a Tokenizer using the Tensorflow library and converted the training sentences into numeric sequences padded with extra zeros.



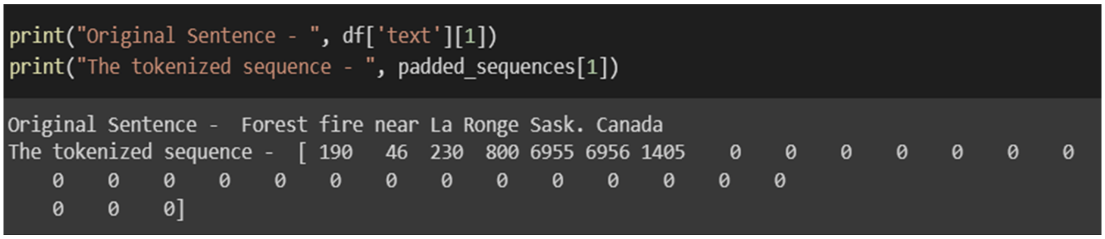
```
# Creating a tokenizer
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_token)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(training_sentences)
padded_sequences = pad_sequences(sequences, padding='post', maxlen=max_len)
```

Here are some of the sentences converted -



```
print("Original Sentence - ", df['text'][0])
print("The tokenized sequence - ", padded_sequences[0])
```

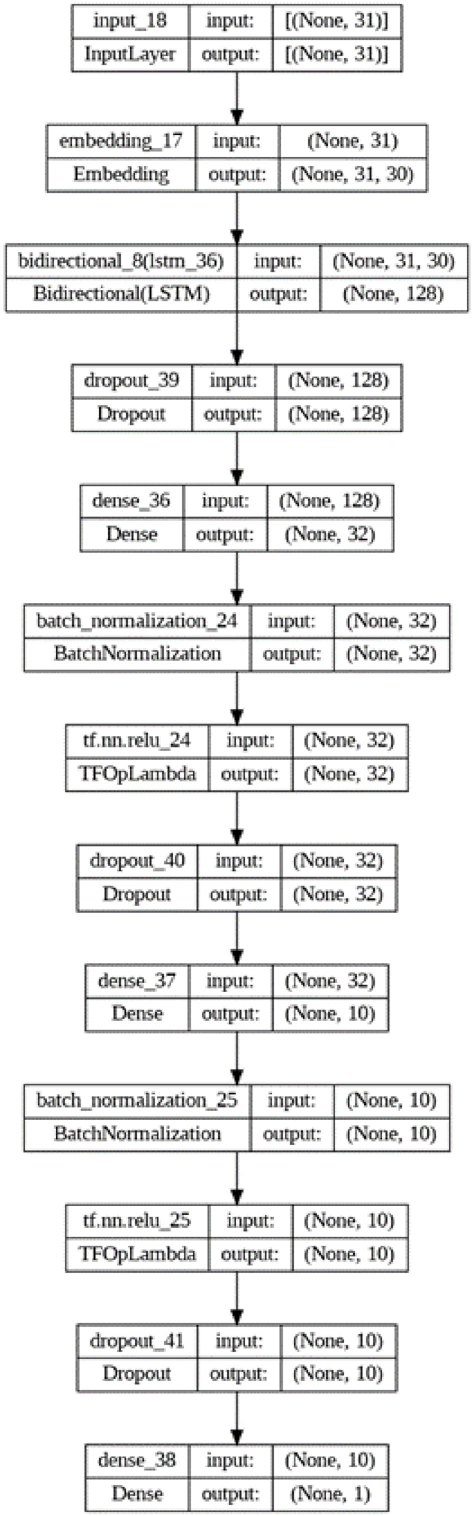
```
Original Sentence -  Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all
The tokenized sequence -  [ 120 4634   25    5  869    9   22  264  139 1620 4635   90   41    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0]
```



```
print("Original Sentence - ", df['text'][1])
print("The tokenized sequence - ", padded_sequences[1])
```

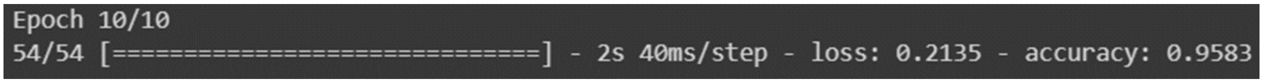
```
Original Sentence -  Forest fire near La Ronge Sask. Canada
The tokenized sequence -  [ 190   46  230  800 6955 6956 1405    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0]
```

# Creating the Model

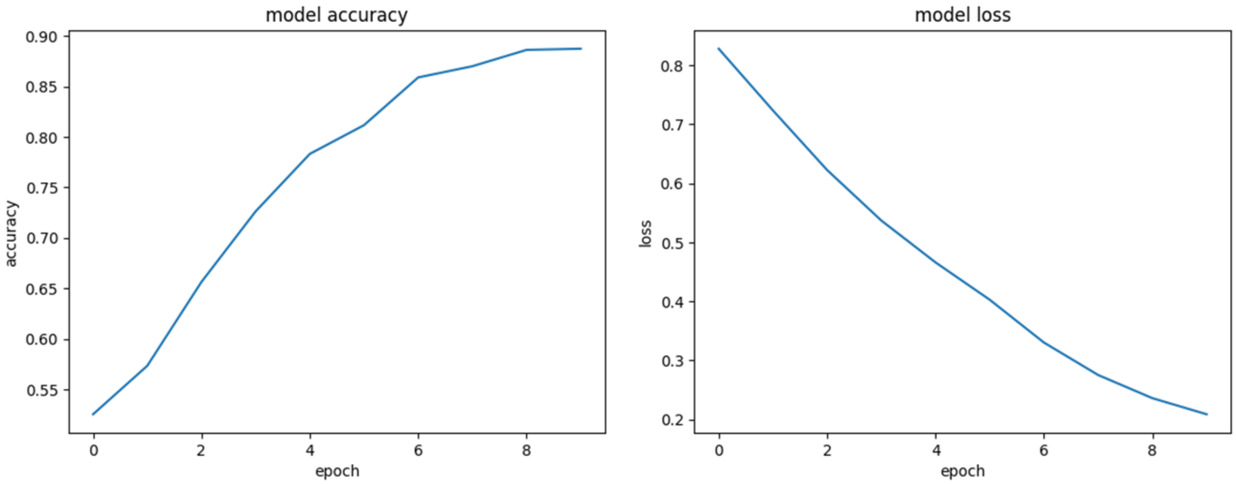

input_18 | input: | [(None, 31)]
InputLayer | output: | [(None, 31)]
embedding_17 | input: | (None, 31)
Embedding | output: | (None, 31, 30)
bidirectional_8(lstm_36) | input: | (None, 31, 30)
Bidirectional(LSTM) | output: | (None, 128)
dropout_39 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_36 | input: | (None, 128)
Dense | output: | (None, 32)
batch_normalization_24 | input: | (None, 32)
BatchNormalization | output: | (None, 32)
tf.nn.relu_24 | input: | (None, 32)
TFOpLambda | output: | (None, 32)
dropout_40 | input: | (None, 32)
Dropout | output: | (None, 32)
dense_37 | input: | (None, 32)
Dense | output: | (None, 10)
batch_normalization_25 | input: | (None, 10)
BatchNormalization | output: | (None, 10)
tf.nn.relu_25 | input: | (None, 10)
TFOpLambda | output: | (None, 10)
dropout_41 | input: | (None, 10)
Dropout | output: | (None, 10)
dense_38 | input: | (None, 10)
Dense | output: | (None, 1)

# Results on Training Data

After training the model on the data for 10 epochs a high accuracy of 95%.

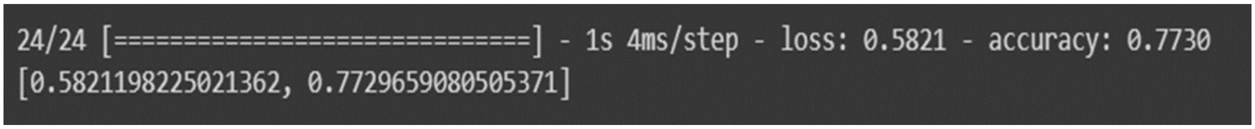


```
Epoch 10/10
54/54 [==============================] - 2s 40ms/step - loss: 0.2135 - accuracy: 0.9583
```



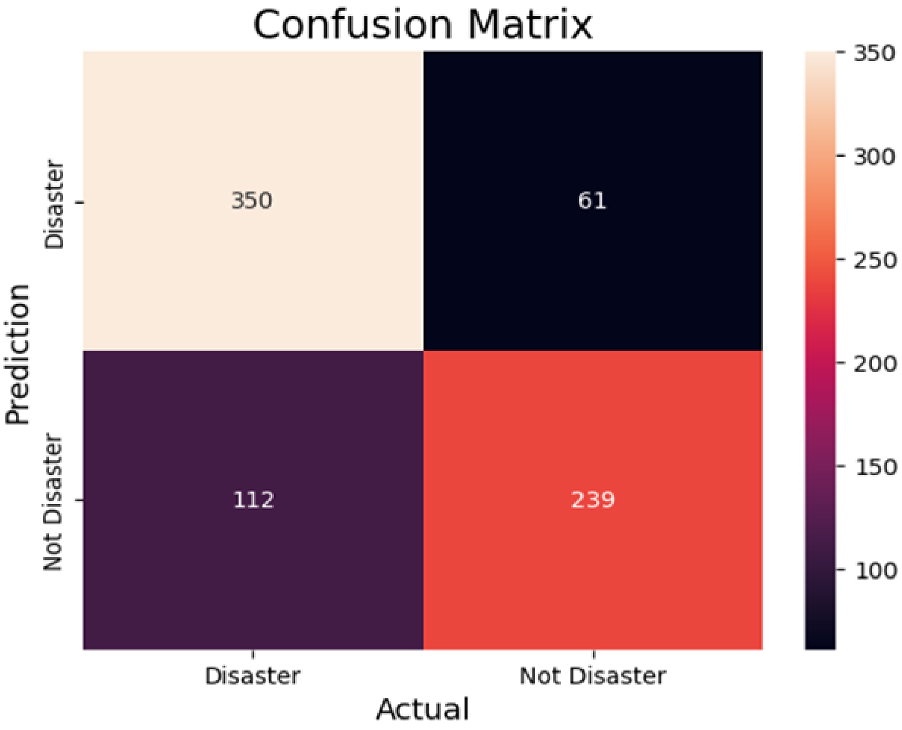
# Results on Testing Data

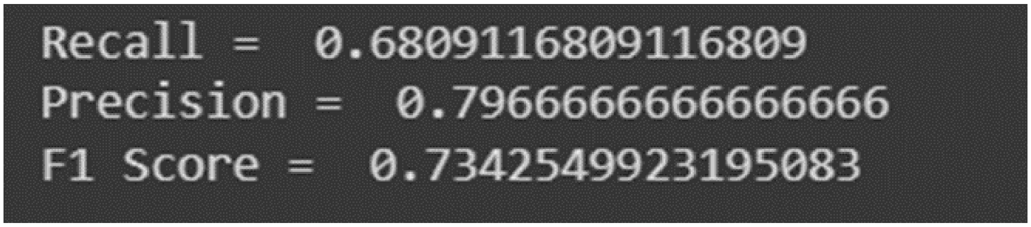
We were able to get an accuracy of around 78% on the testing data.



```
24/24 [==============================] - 1s 4ms/step - loss: 0.5821 - accuracy: 0.7730
[0.5821198225021362, 0.7729659080505371]
```

Some other metrics are also summarized below:





```
Recall =  0.6809116809116809
Precision =  0.7966666666666666
F1 Score =  0.7342549923195083
```

# Conclusion

- We studied the concepts of regression and binary classification. Then we studied some concepts of Natural Language Processing including RNNs, GRUs and LSTMs, and word embeddings.

- We trained a sequential model using Tensorflow on the training data and achieved a high accuracy of around 95%.

- On the test data also, we were able to get a good accuracy of around 78% which is much higher than benchmark accuracy.

# Improvements and Future Scope

- We can use different tokenizers and some pre-trained tokenizers which can definitely improve the results.

- We can clearly see that the model is undergoing overfitting, so we can use some techniques like early stopping, removing some layers, some regularization methods etc. to remove overfitting.

- We can use pre-trained word embeddings like GLove, Word2Vec, etc.

- Some state-of-the-art models like the BERT transformer can be used.

# Bibliography

1. https://towardsdatascience.com/

2. www.sciencedirect.com

3. www.analyticsvidhya.com

4. https://www.coursera.org/learn/neural-networks-deep-learning/home/welcome

5. https://www.coursera.org/learn/nlp-sequence-models/home/welcome