**FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSING (NLP)**



**Name: Prarthana B R**

**Reg.No: 513521104037**

**Department: CSE**

**Year: III**

**NM ID: au513521104037**

**E-Mail: prarthanaravi1330@gmail.com**

**Phase 4 (Development Part -2)**

**INTRODUCTION:**

The spreading of fake news causes many problems in the society. It easily deceives people and leads to confusion among them. It has the ability to cause a lot of social and national damage with destructive impacts. Sometimes it gets very difficult to know if the news is genuine or fake. Therefore it is very necessary to detect if the news is fake or not.

**Natural Language Processing (NLP)** and deep learning techniques are powerful tools for fake news detection. We tend to use **LSTM approach** for this project.

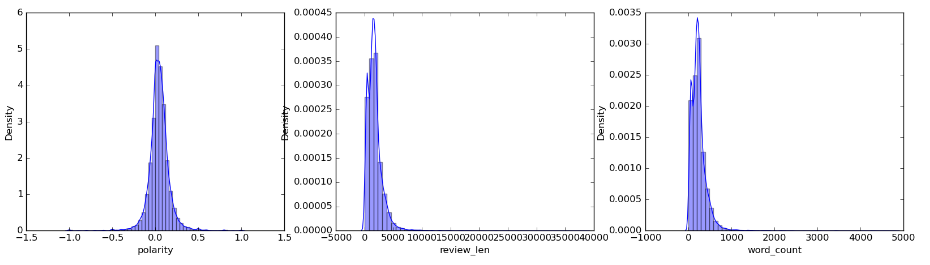
In the continuation of the previous documentation AI\_Phase3, we will be looking forward on the second phase of the development of the project. In this phase, we will **perform Feature Engineering, Text pre-processing, Feature Extraction, building, fitting the Model and evaluating the features.** It will also provide inference about the model training and testing. The following steps are followed in this development part-2.

1. **Feature Engineering:**

* Feature Engineering is the process of taking raw data and transforming them into certain features that help in creating a predictive model using standard modelling methods.
* It is also a form of Exploratory Data Analysis.
* This includes :
  1. Highlighting the features of dataset
  2. N-Gram Analysis (Unigram, Bigram)
  3. WordCloud
  4. **Highlighting the features of dataset:**

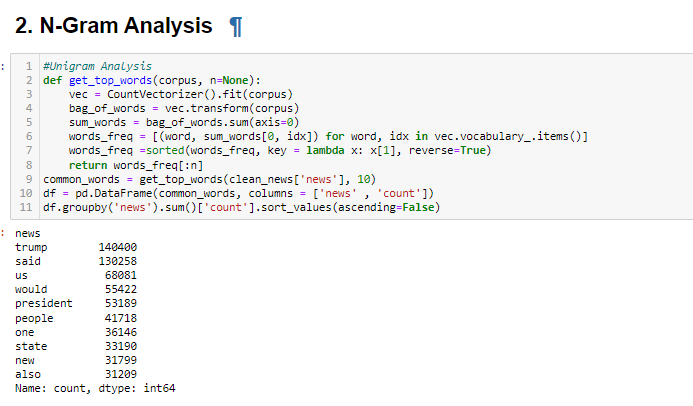
****

* On running this we get output as,

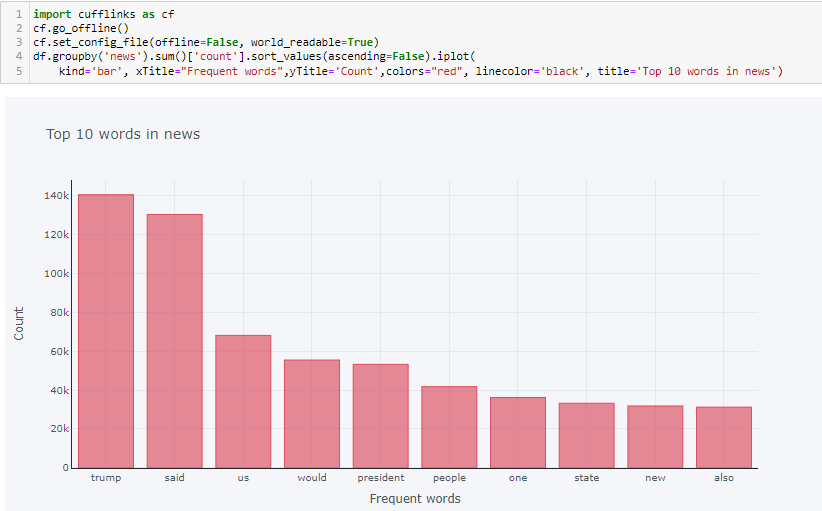


* From these we get insights about:
* **Polarity:** The measure which signifies the sentiment of the news.
* **Review length:** Length of the news (number of letters and spaces).
* **Word Count:** Number of words in the news.
* We can infer that most of the polarity are neutral, neither it shows some con nor much pro.
  1. **N-Gram Analysis:**

1. **Unigram Analysis:**
   * Let's look at the top 10 words from the news which could give us a brief idea on what news is popular in our dataset.

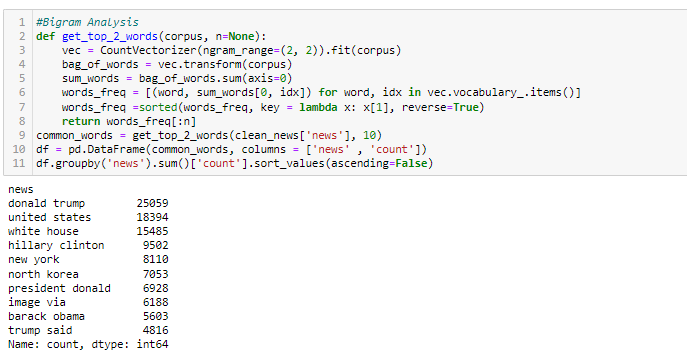
****

* + Let's look at the top 10 words graphically from the news dataset using cufflinks that gives a detailed description about the features associated to it also.

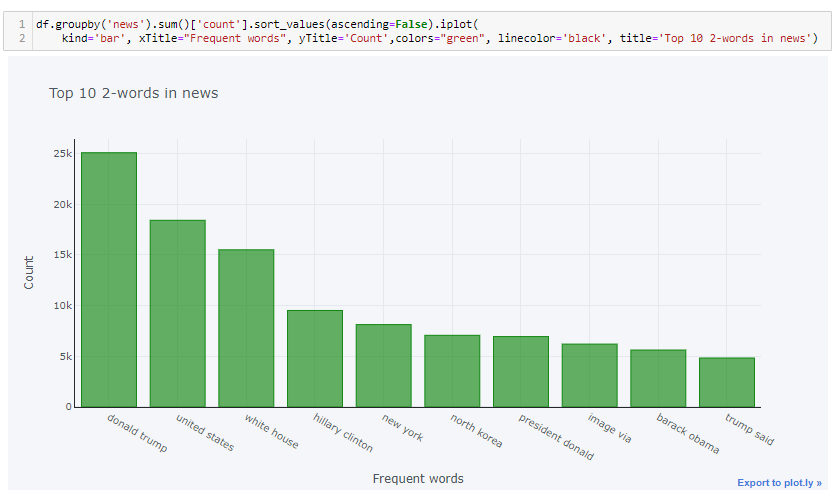
****

* + It is clear that the news is all about Trump and revolves around US, President etc.,

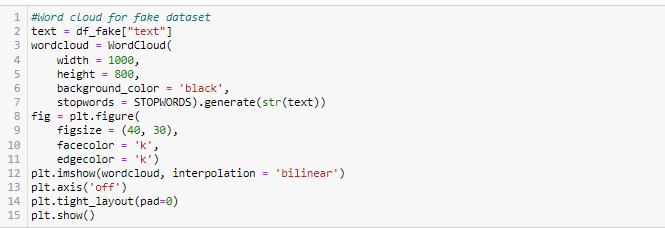
1. **Bigram Analysis:**
   * Now let's expand our search to top 10 2-words from the news.

****

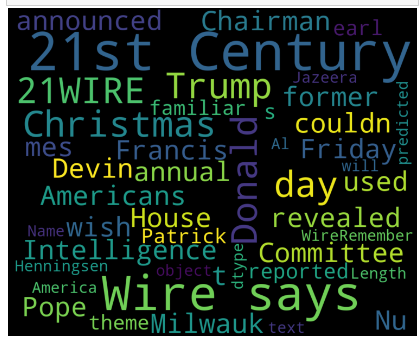
* + Let's look at the top 10 2-words graphically from the news dataset using cufflinks that gives a detailed description about the features associated to it also.

****

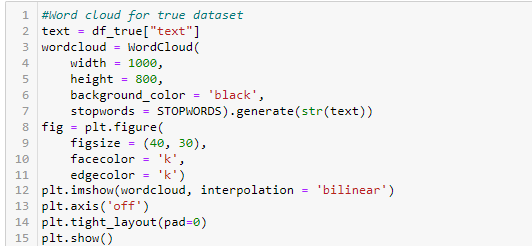
* + As expected, the news revolves around Donald Trump and United states majorly, we can infer that the dataset is quite biased.
  + It also describes about politicians and events in NewYork.
  1. **Word-Cloud:**
     + - It is the pictorial representation of familiar words from the dataset.
       - Word Cloud code for texts from Fake news dataset:



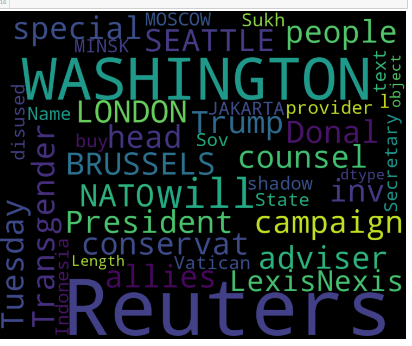
* + - * On running this, we get:



* + - * We can infer that most of the news revolves around Trump as mentioned earlier.
      * Word-Cloud code for texts from True news dataset:



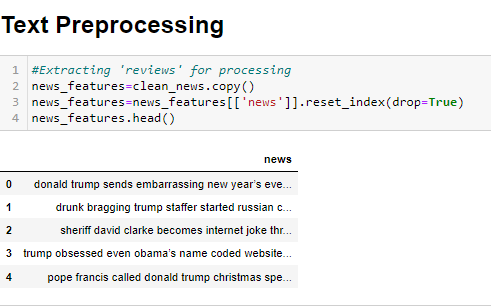
* + - * On running this, we get:



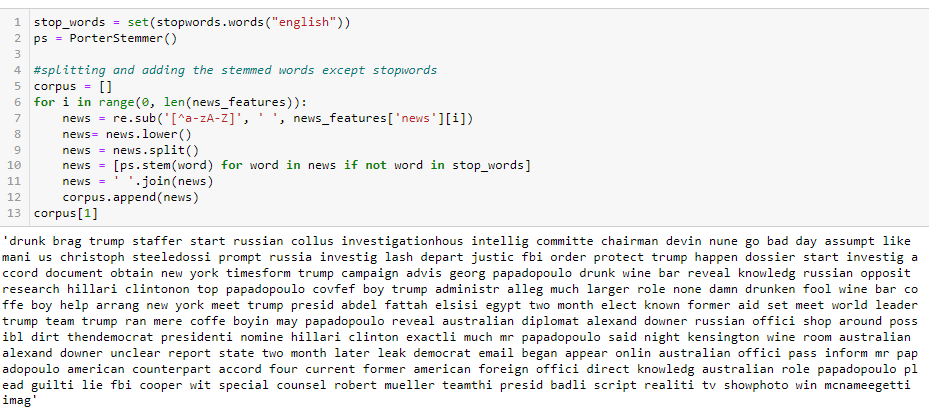
* + - * We can infer that most of the news is from Reuters and Washington.

1. **Text Pre-processing:**

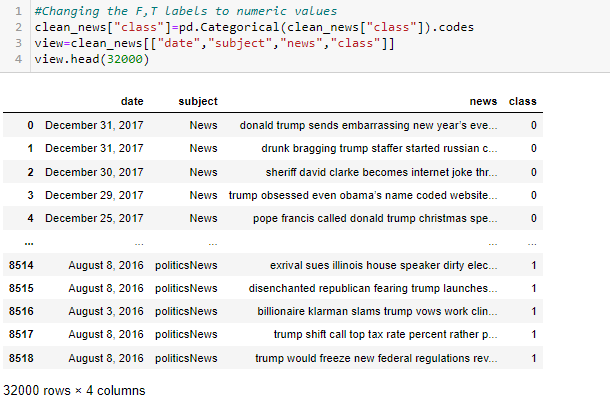
* We extract certain features only from the news column of clean\_news as:



* Then we perform Stemming. Stemming is a method of deriving root word from the inflected word. Here we extract the reviews and convert the words in reviews to its root word. For example, **Going -> go, Finally -> fina**
* It is given as:



* As, pandas cannot process strings for the classification reports, we convert the labels of the class to numeric values as 0, 1. False label is converted to 0 and True label is converted to 1 as:



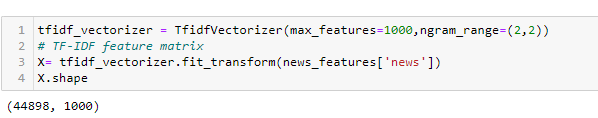
1. **Feature Extraction:**

It involves 2 steps namely:

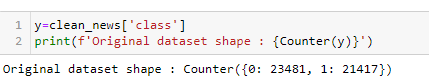
* 1. Word Embedding by TF-IDF
  2. One hot Representation
  3. Padding

**3.1) Word Embedding by TF-IDF:**

* Word embedding is a method of converting the text to corresponding numeric values for giving a semantic meaning.
* TF-IDF is also a method for embedding the words.
* TF-IDF stands for “Term Frequency — Inverse Document Frequency”. This is a technique to quantify a word in document, we generally compute a weight to each word which signifies the importance of the word in the document and corpus.
* This method is a widely used technique in Information Retrieval and Text Mining.
* Here we are splitting as bigram (two words) and consider their combined weight. Also we are taking only the top 1000 words from the news.

****

* As we have considered 1000 words, we can confirm that we have 1000 columns from the shape.

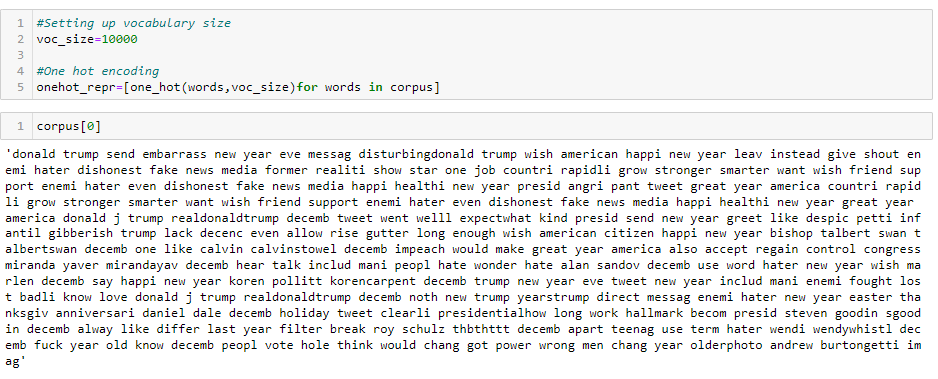


* As we have considered 1000 words, the shape remains balanced yet. So let’s proceed further.
* Using train test split function we are splitting the dataset into 80:20 ratio for train and test set respectively.



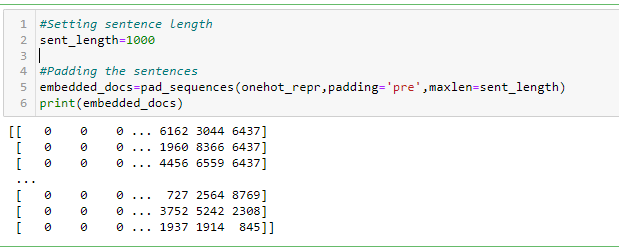
**3.2) One hot Representation:**

* We will be one hot encoding the sentences in the corpus for processing embedding layers in the model.
* While one hot encodes the words in sentences will that take the index from the vocabulary size.
* Let's fix the vocabulary size to 10000 and view the result of one hot encoding.

****

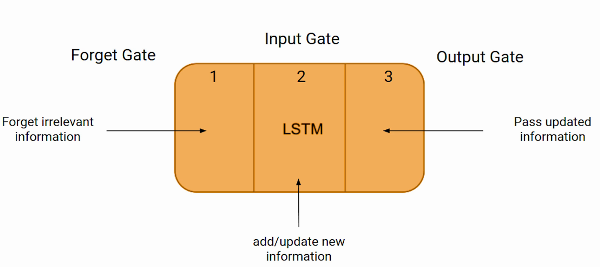
**3.3) Padding:**

* All the neural networks required to have inputs that have the same shape and size.
* However, when we pre-process and use the texts as inputs for our LSTM model, not all the sentences have the same length. In other words, naturally, some of the sentences are longer or shorter.
* We need to have the inputs with the same size, this is where the padding is necessary. Here we take the common length as 1000 and perform padding using pad\_sequence() function .
* Also we are going to 'pre' pad so that zeros are added before the sentences to make the sentence of equal length.
* Padding is given as:

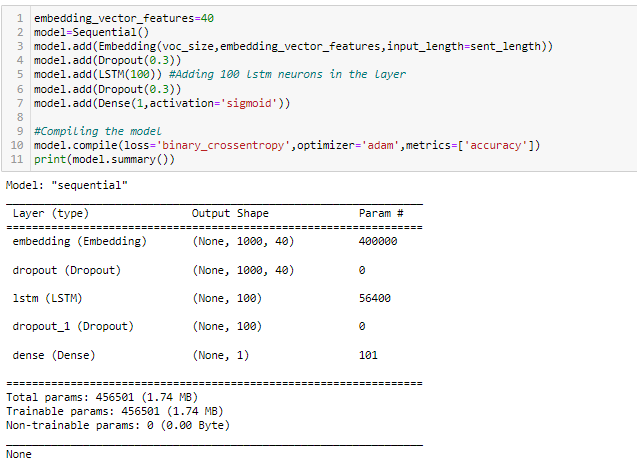
****

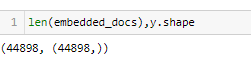
1. **Model Building:**

* We can use any of the models like Logistic Regression, Random Forest Classifier etc.,
* But here, we use neural network to predict whether the given news is fake or not.
* We aren't going to use normal neural networks like ANN to classify but **LSTM (Long Short Term Memory)** which helps in containing sequence information.
* **Long Short-Term Memory (LSTM) networks** are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.
* This is a behaviour required in complex problem domains like machine translation, speech recognition, and more.
* At first we are going to develop the base model and compile it.
* The **first layer** will be the **embedding layer** which has the input of vocabulary size, vector features and sentence length.
* Later we add 30% **dropout layer** to prevent over-fitting and the LSTM layer which has 100 neurons in the layer.
* In **final layer** we use **sigmoid activation function**. Later we compile the model using adam optimizer and binary cross entropy as loss function since we have only two outputs.
* LSTM works by remembering only the important sequence of words and forgets the insignificant words that don’t add value in the prediction.



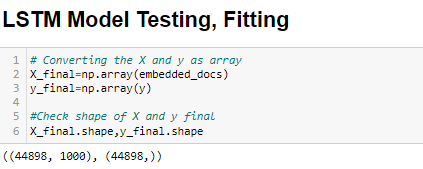
* By changing the number of neurons in the LSTM layer, we can improve the accuracy of the model.
* It is given as:



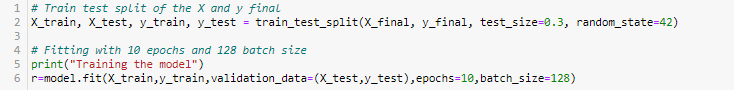


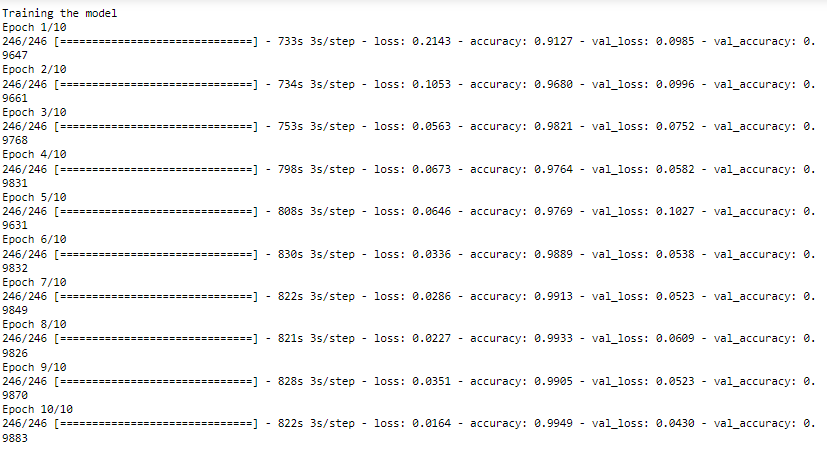
1. **Model Fitting and Training:**

* Before fitting to the model, let's consider the padded embedded object as X and y as y itself and convert them into an array.

****

* Let's split our new X and y variable into train and test and proceed with fitting the model to the data. We have considered 10 epochs and 128 as batch size.
* The number of epochs and batch size can be varied to get better results.
* The training of our model is given as**:**

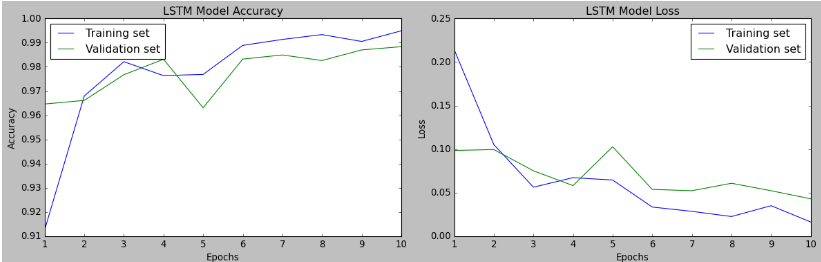
****

****

* We can increase the batch size for faster training of the model.
* On plotting the graph for the training result, we can infer the model’s fitting.



* The graph is given as:



* From the graph, we can see that training and validation accuracy gradually increases with little variations in their lines indicating a good movement.
* Likewise, we can see the training and validation loss gradually decreases from a point with little variations between them indicating a good movement.
* Hence the model turns out to be a good fit model, hence we can further proceed.
* If incase, the model overfits or underfits, we are supposed to train them again by altering the size of batch and number of epochs until they fit better.

1. **Model Evaluation:**

This involves 4 metrics namely:

* 1. Confusion Matrix
  2. Accuracy
  3. Classification Report
  4. ROC-AUC Curve

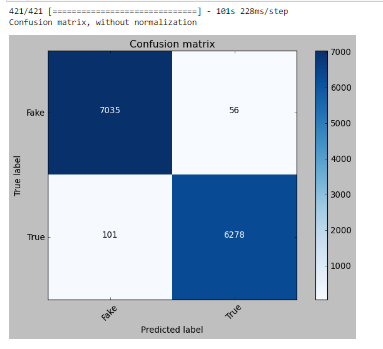
**6.1) Confusion Matrix:**

* It is a table that is used in classification problems to assess where errors in the model were made.
* The rows represent the actual classes the outcomes should have been.
* While the columns represent the predictions we have made.
* Using this table it is easy to see which predictions are wrong.
* The Confusion Matrix has four different quadrants:
  + 1. True Positive (TP): It is the total counts having both predicted and actual values are true.
    2. True Negative (TN): It is the total counts having both predicted and actual values are false.
    3. False Positive (FP): It is the total counts having prediction is true while actually are false.
    4. False Negative (FN): It is the total counts having prediction is false, while actually, it is true.



* The Confusion Matrix for our model is:





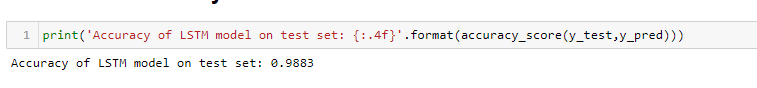
* The diagonal elements (7035+6278), they are correctly predicted records and rest are incorrectly classified by the algorithm.
* Also the other diagonal values of True Negative and False Positive are low.
* This shows that, our model has done well.

**6.2) Accuracy:**

* Accuracy measures how often the model is correct.
* The accuracy is given by:

**Accuracy = (True Positive + True Negative) / Total Predictions**

* The accuracy of our model is:



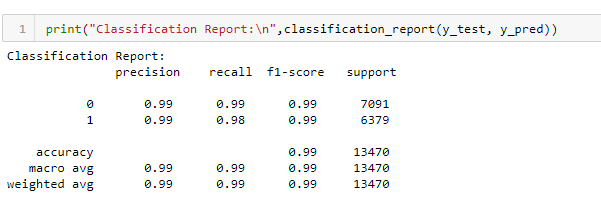
* The accuracy of our model is **98.83%.**

**6.3) Classification Report:**

* This is the summary of the quality of classification made by the constructed ML model.
* It comprises mainly 5 columns and (N+3) rows.
* The first column is the class label’s name and followed by Precision, Recall, F1-score, and Support.
* N rows are for N class labels and other three rows are for accuracy, macro average, and weighted average.
* The columns are:

1. **Precision:** The precision tells us the **accuracy** of positive predictions.
2. **Recall:** It tells us the **fraction** of correctly identified positive predictions.
3. **F1 Score:** The f1-score, or F measure, measures **precision and recall** at the same time by finding the harmonic mean of the two values. This score is useful when you have opposite scores coming from precision and recall.
4. **Support:** The support is the number of occurrences of each class in your y\_test.

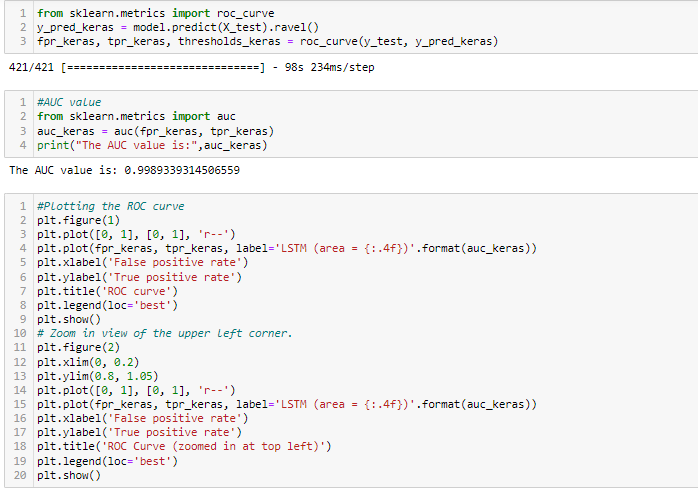
* The classification report of our model is:



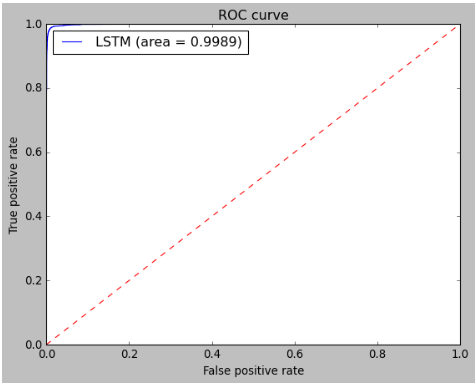
* It gives an average around 99% which is good.

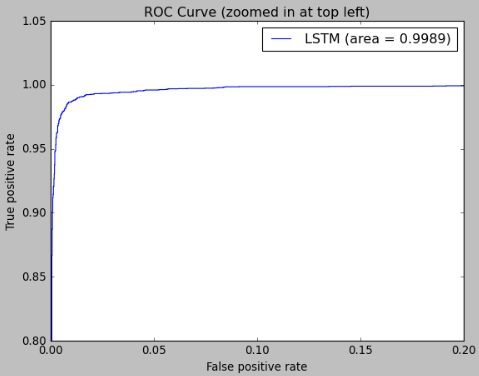
**6.4) ROC-AUC Curve:**

* An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters namely, **True Positive Rate and False Positive Rate.**
* **AUC** stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).
* **The ROC-AUC for our model is:**

****

* **The ROC-AUC curve for our model is:**

****

****