**Machine Learning Project Report**

**Fake News Detection**

**Submitted to:** Sir Usman Shahid

**Member 1**

Name: Aneeq Asghar

Roll number: 17L-4525

Section: EE-7A

**Member 2**

Name: Salman Shah

Roll number: 17L-4449

Section: EE-7A

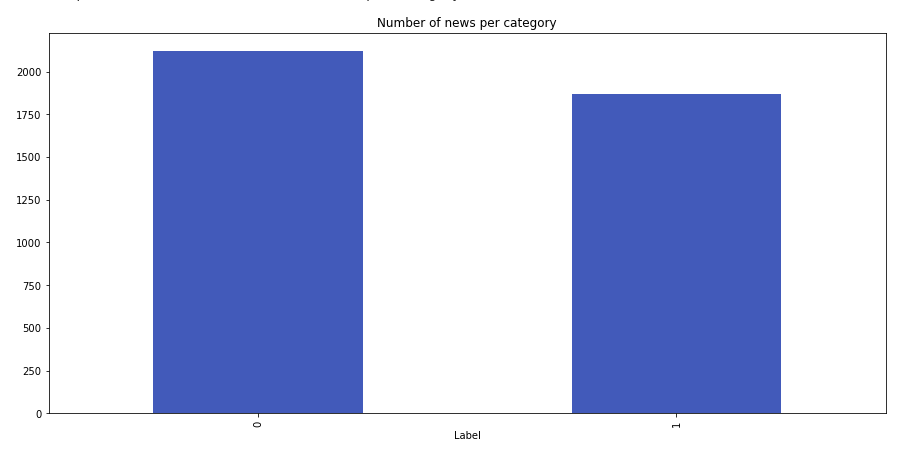
***Introduction:***

The machine learning project we selected is on Fake News detection. We need to classify the news from the given dataset data.csv in which there are URLs, headlines, body, and labels. We have to analyze the data which is in the form of words and sentences so we have to perform some kind of sentiment analysis in the python code using a simple machine learning algorithm and deep learning. There are three phases of this project. In phase 1 of the project, we predicted the Fake News using a simple machine learning model. The machine learning model used to predict the fake news in phase one of the project is on Logistic regression. Now, in the second phase of the project, we used deep learning to predict the news. We had to map alphabetic data to numerical values, also we had to perform NLP (natural language processing) on an overview of the news to get its impact on the label.

**Data Acquisition:**

We acquired the dataset from (<https://www.kaggle.com/jruvika/fake-news-detection>) of Fake news dataset from Kaggle. The dataset was a CSV file in which each news was categorized under columns of URLs, Body, Headline, and Label respectively. The dataset had 4009 rows and 4 columns.

Visualizing the data is an important part before applying any Machine Learning Algorithm and we too visualized the dataset by plotting a bar chart while differentiating data as fake or true where zero is the label for fake news and one is the label for a news being true.



**Data Preprocessing:**

Before applying any Machine Learning algorithm we firstly have to preprocess the data.

There are three main phases of preprocessing the data

1. Data cleaning
2. Data munging: a methodology for taking data from unusable forms to the new levels of structure and quality required by modern analytics processes and consumers.
3. Map alphabetical data to numerical values

We cleaned the data in each phase differently depending upon the requirements of the project.

Concatenating the Body and Headline of each news was done as the first part of data pre-processing. We removed all the NAN (Not a Number) rows and columns and reset the index of the dataset. We downloaded and imported all the required libraries for data cleansing such as pandas, regular expression library, Tfidf vectorizer from Sklearn to convert the alphabets to numerical values in phase one of the project whereas, I used Tokenizer from Keras Tensorflow in phase 2 of the project for encoding words. Padding the data was another important part of data preprocessing. We used pad\_sequences from Tensorflow Keras in phase 2 for padding each individual news’s concatenated body and headline to a fixed length to apply word embedding to each of the news whereas, we did not pad the news in phase 1 of the project.

**Section 1**

**Machine Learning:**

In the first phase, we were using the machine learning approach and classified the data using Logistic regression after preprocessing the data. We start the prediction through the body of the news. Then by importing the classifier we directly classified the data and made the confusion matrix using Sklearn library to classify all the true positive, true negative, false positive, and false negative predicted through our classifier. Each quadrant of the Confusion Matrix provides us with the following:

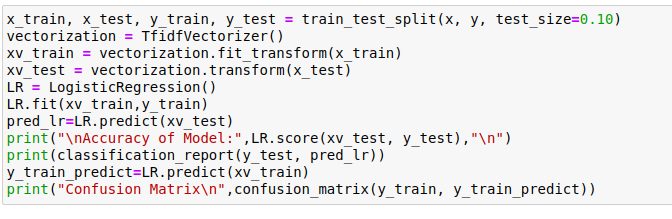


We also made a classification report to predict the accuracy, precision, and F1 score of the model. Now, we split the dataset into five different sizes of training and test set to check that which of the following gives the maximum accuracy for the model we built. The exact percentage through which the data was split is shown below:

1. 75% training set and 25% test set.
2. 85% training set and 15% test set.
3. 60% training set and 40% test set.
4. 90% training set and 10% test set.
5. 70% training set and 30% test set.

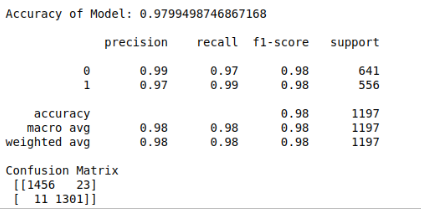
The code for splitting the dataset into 90% train dataset and 10% test dataset by setting the test\_size to 0.1. The Classification Model used in this experiment is the Logistic Regression for all of the five splits. Logistic Regression is applied using the inbuilt Logistic Regression from Sklearn.linear\_model library.

Furthermore, the accuracy and confusion matrix for each of the splits is calculated using the code given below.

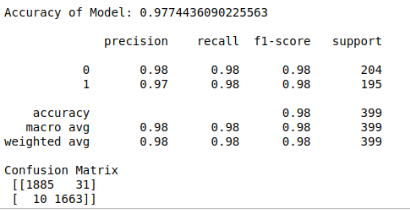


The accuracy, precision, F1- score and Confusion matrix for each of the Test Case is given below:

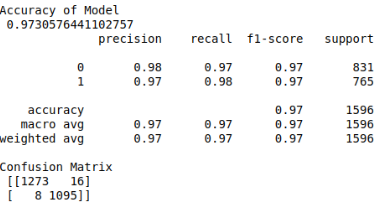




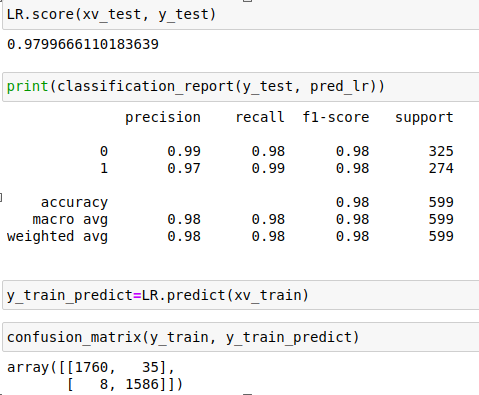




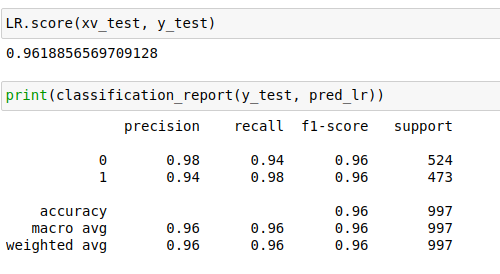


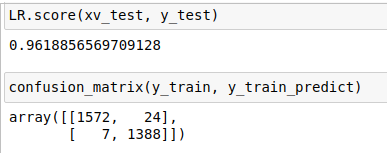












Test Cases for Machine Learning Model:

|  |  |  |
| --- | --- | --- |
| **Case Number** | **Test Case** | **Accuracy** |
| 1 | 75% training set and 25% test set. | 96.1 % |
| 2 | 85% training set and 15% test set. | 97.996 % |
| 3 | 60% training set and 40% test set. | 97.3 % |
| 4 | 90% training set and 10% test set. | 97.7 % |
| 5 | 70% training set and 30% test set. | 97.994 % |

As we have seen the accuracy of each of the above-mentioned test cases. We have selected the train-test split with the maximum accuracy which is 85% training set and 15% test set. It has a maximum accuracy of 97.9966%.

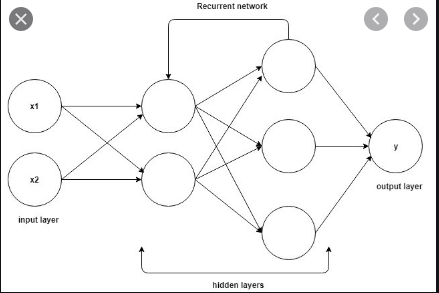
**Phase 2**

**Deep Learning :**

The deep learning algorithms to predict news as we have to analyze the words to predict the news, by converting the words into tokens :

* Recurrent Neural Networks

It uses the multilayer neural network but the issue over here is that it has a vanishing gradient problem as we will be using backpropagation by derivating the term many times. By having a large number of derivatives in the process we are not able to reach the local minima and a vanishing gradient issue occurs.

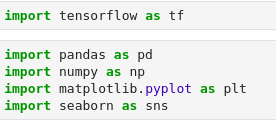


* Long Short Term Memory

The issue of vanishing gradient is resolved by using Long Short-Term Memory which remembers the previous output of the nodes and sends it as an input to the next. It also stores the output of this function as an input to the node as feedback to improve the performance and accuracy of the model. Such as, if the word is sent from a sentence to the hidden layer of the neuron, then the weight is assigned to it and the word is preprocessed and the previous output is also given to the layer of neural network and the sequence is maintained.



The command shown above is used to convert the dataset to a list.



1. Tensorflow is used for deep learning and neural networks.
2. Pandas library is imported to import the dataset.
3. Numpy library is used for mathematical library operations
4. Seaborn and matplotlib library is used for data visualization in plotting images and graphs.

The basic procedure for applying deep learning in our model is to first clean the dataset and convert the words to tokens for statistical computation and prediction. The dataset is cleaned using the

**Cleaning the dataset:**

It is done using by firstly removing all the Not a Number columns and rows in the dataset by using the dropna command.

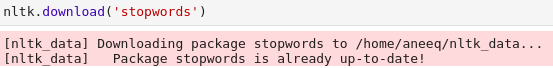
Secondly, we remove the URL column from the dataset as we do not need it to predict the label of the news:

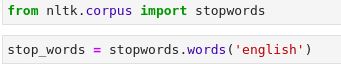


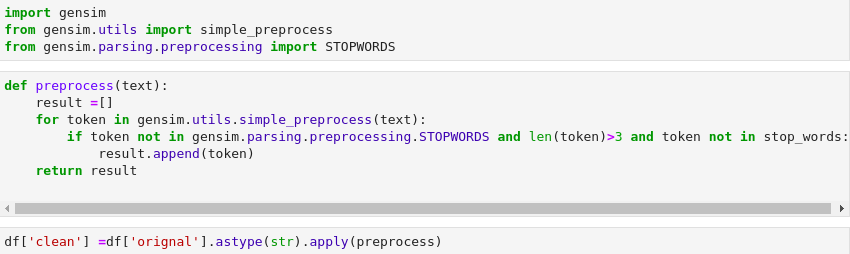
Thirdly, we combined both the Headline and the Body of the news through which we will do the computations as shown below:



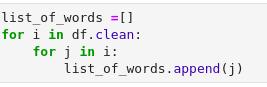
Fourthly, we removed the English stopwords from the dataset which aren’t of importance while predicting the label of the news. We are removing these stopwords to reduce the size of the data and to reduce the load on the PC. This is done by using the following commands:







Now, the data frame clean is changed to a tuple by using a for loop and appending in the list of words:



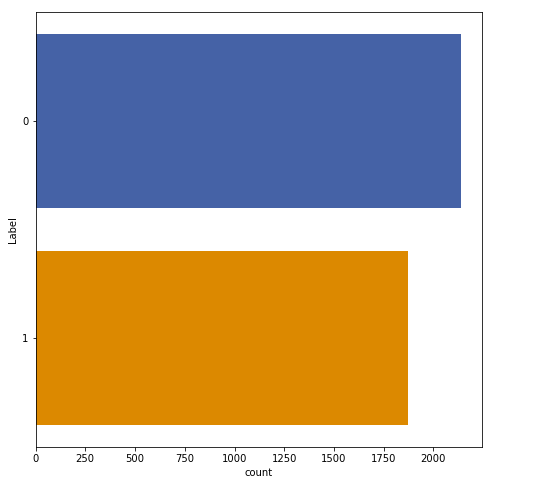
These bag of words are converted to a continuous list by appending a space after each line as shown below:



**Data Visualization:**

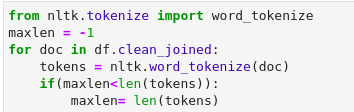
Data is classified as fake or true based on the label on it. It is visualized by using the following commands:



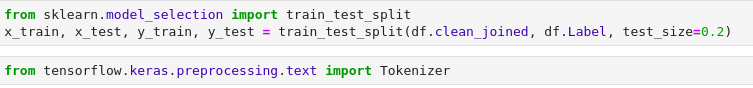


**Convert words to tokens and padding:**

Words are converted to tokens using the tokenizer and the maximum length of an individual line or row is computed. The maximum length is used to pad the lines by appending zeros after the length of the individual line. Data is separated into training and test set to check the accuracy of the model where 20% of the data is in the test set whereas, 80% of the data is in the training set. The tokenizer is used to convert the data into tokens or numbers replaced for each word in the data. It also removes punctuations and URL from the data. Tokenizer is used to substitute a randomly generated identifier in order to prevent unauthorized access.







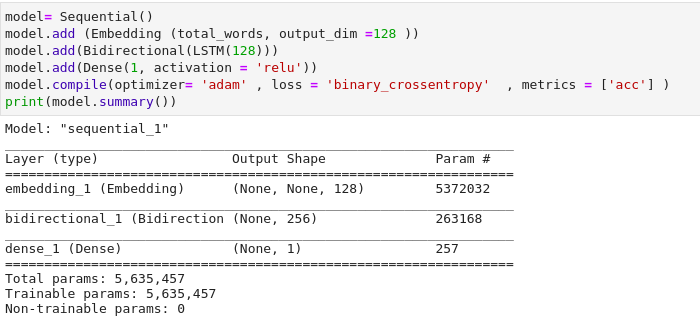


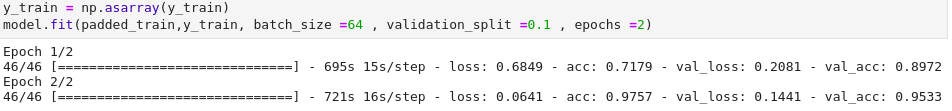


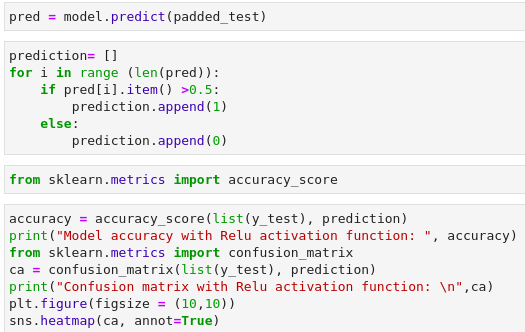
The last thing we will do is to implement the Deep learning algorithm LSTM in it.

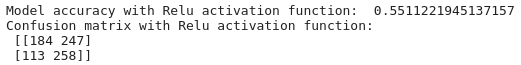
We will do the word embedding. It is done to boost the generalization and performance of LSTM. Here, we will form a sequential model and use a bidirectional LSTM layer with 128 units as we have 128 output dimensions of the word embedding. The summary of the model is shown below:

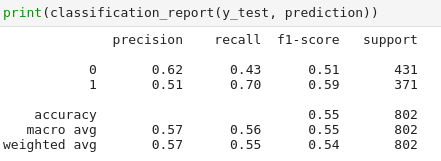
**Iteration 1 of model 1:**



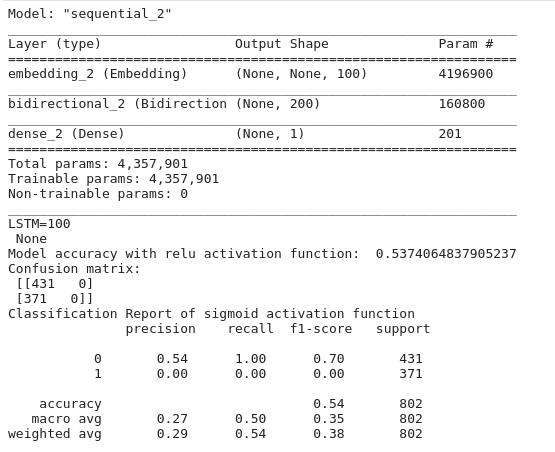




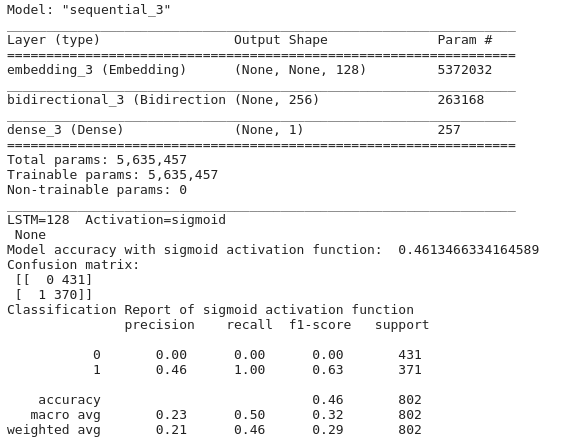




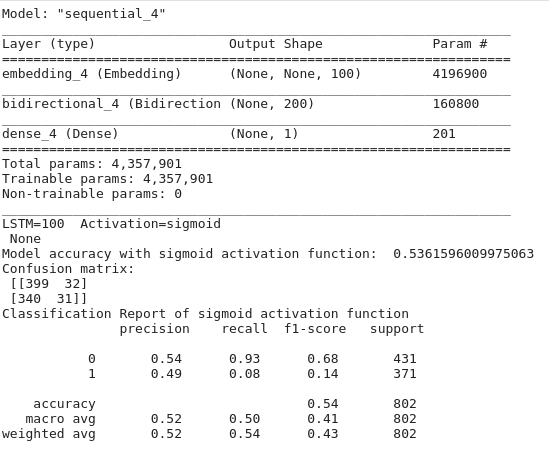
**ITERATION 2 of MODEL 1:**



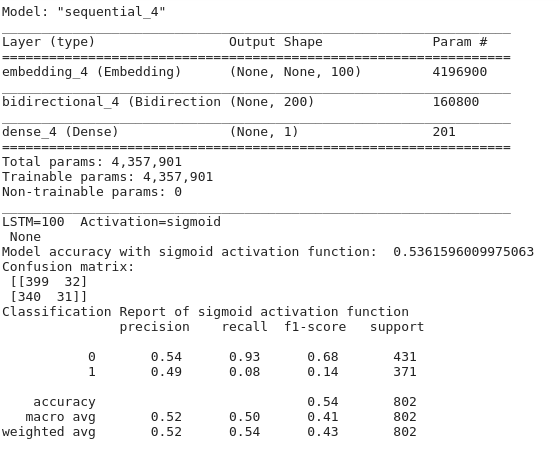
**ITERATION 1 of MODEL 2:**



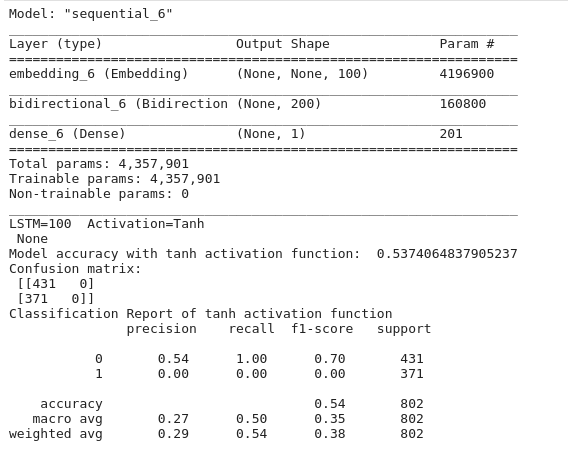
**ITERATION 2 OF MODEL 2:**



**ITERATION 1 OF MODEL 3:**



**ITERATION 2 OF MODEL 3:**



**TEST CASES:**

**Deep learning RNN LSTM APPROACH:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case No.** | **MODEL** | **Iteration** | **Accuracy** |
| 1 | 1 (Relu Activation Function) | 1 (128 LSTM units) | 55% ~97.57% with the model |
| 2 | 1 (Relu Activation Function) | 2 (100 LSTM units) | 96% |
| 3 | 2 (Sigmoid Activation Function) | 1 (128 LSTM units) | 97% |
| 4 | 2 (Sigmoid Activation Function) | 2 (100 LSTM units) | 96% |
| 5 | 3 (Hyperbolic Tangent Function) | 1 (128 LSTM units) | 97% |
| 6 | 3 (Hyperbolic Tangent Function) | 2 (100 LSTM units) | 95% |

Test case 1 is Model 1 with iteration 1 selected due to the highest accuracy which is 98% when the model is fitted to the dataset. Hence, a Rectified Linear Unit with an LSTM of 128 units is selected.

**Issues:**

Fitting the model onto the dataset took a lot of time and my PC got stuck several times hence, I could not do the number of iterations required to select a model to predict the result.

**Improvements:**

We have not utilized the URLs columns but we could have utilized it in the deep learning part but we are just using the body and headline to predict the news, but still model gives 97% of the accuracy.

**Conclusion:**

Accuracies for using a Simple Machine Learning model to classify the news were quite high and using 85% training data and 15% test data of the whole dataset gives us the highest accuracy which is an astonishing 97.99%. By using deep learning we also achieved a record 97% accuracy. Computing for 128 LSTM units gives a better accuracy rather than 100 LSTM iterations. By using an Embedding layer as the first hidden layer, Rectified Linear Unit activation function as the second layer, and Long Term Short Memory with 128 units, we achieved the highest accuracy of 97.57%.