**CHAPTER FOUR**

**MODEL IMPLEMENTATION, RESULT AND DISCUSSION**

**4.1 introduction**

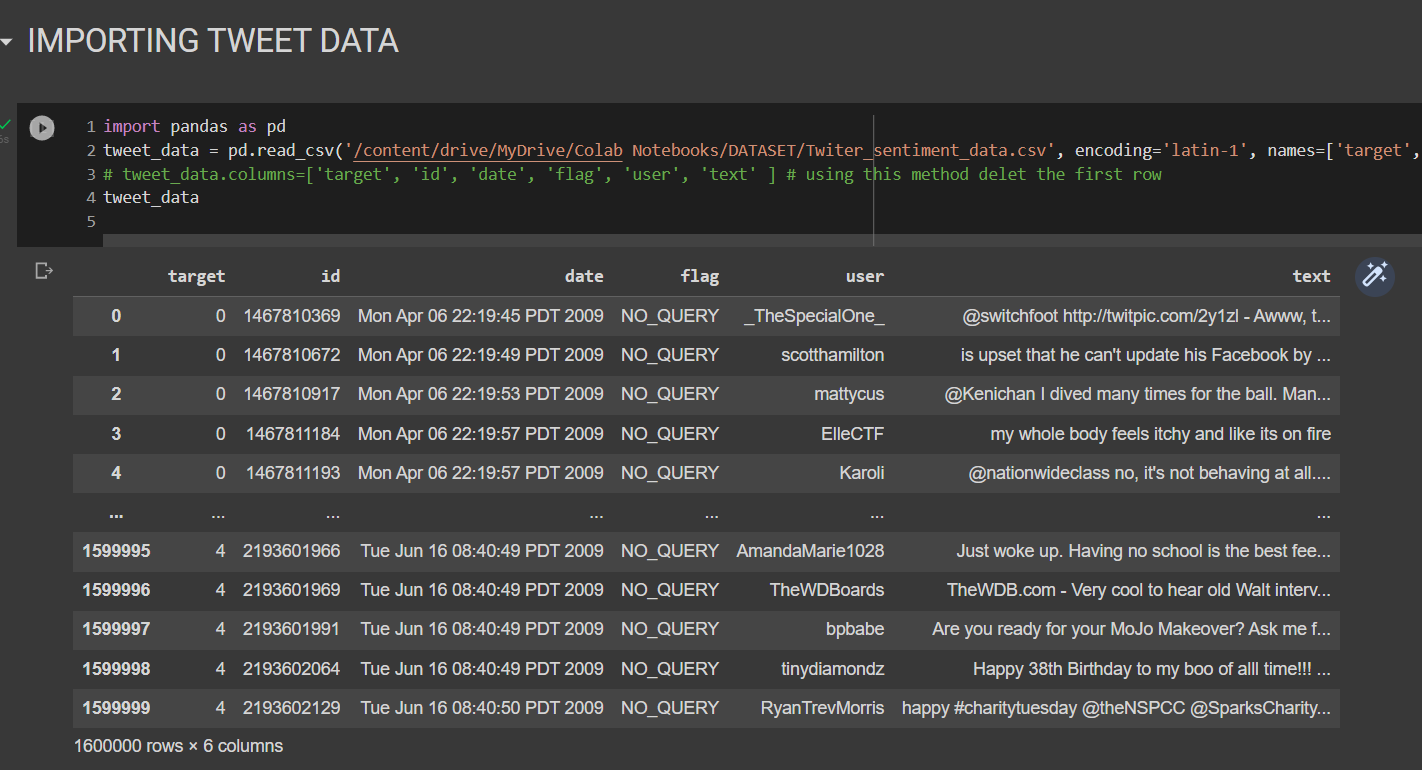
This section illustrates with figures and explanations on various stages taking for achieving the proposed model. This various stage includes, data importing, data exploration, feature extractions approach, data preprocessing and cleaning, Vectorization of textual data point, Model building, training and evaluation. Then the chapter is concluded with the discussion of result using based on the outcome of various performance metrics.

**4.2 Data import and Exploration (EDA)**

This subsection explain how data is imported along with various data exploration carried out on the dataset. this is essential to have a better understanding about the dataset.

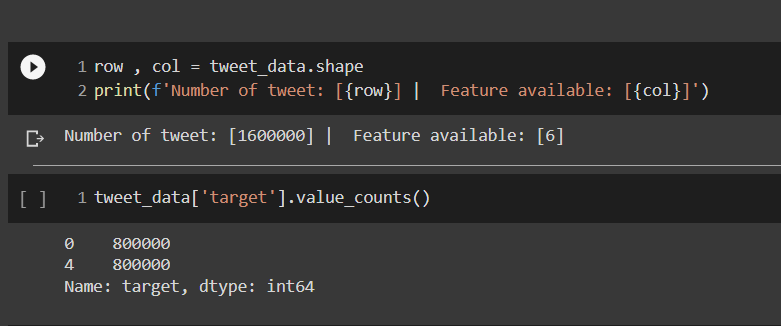
The Sentiment 140 dataset with 1.6 million tweets gotten from Kaggle is download and upload to the google drive for easy access by the Google Collaboratory. Communication between the google Collaboratory and google drive is achieved using the mounting of drives in Google Collaboratory. Then data import is done using pandas in python by specifying dataset path in the Google Drive. The figure 1 shows the code snippet of how the dataset is imported into the google collaborator environment.

Considering the figure 1, the panda framework is imported using the ‘import’ statement and the actual ‘CSV’ file is imported using the ‘*read\_csv’* helper method. The data set is visualized in table format with the help panda framework for easy glance at the dataset, its shown in figure 1 that the dataset consists of ‘target’, ‘id’, ‘data’, ‘flag’, ‘user’, and ‘text’ columns. Furthermore, various exploration is carryout on the dataset for better inside, such as knowing the number of datapoint, feature available, essential features, distinct classification class and the like.



**Figure 1. Importing Tweet Sentiment140 dataset with 1.6 million tweets**

Figure 2. shows that the sentiment140 dataset consist of 1.6 million tweet and 6 features (‘target’, ‘id’, ‘data’, ‘flag’, ‘user’, and ‘text’), based on the **target** they are two possible class which is ‘0’ (negative sentiment) and ‘4’ (Positive sentiment). The exploration carried out show that both the classes contain equal number of datapoint has denoted in figure2 and figure3.



**Figure 2. Exploratory Data Analysis (EDA)**

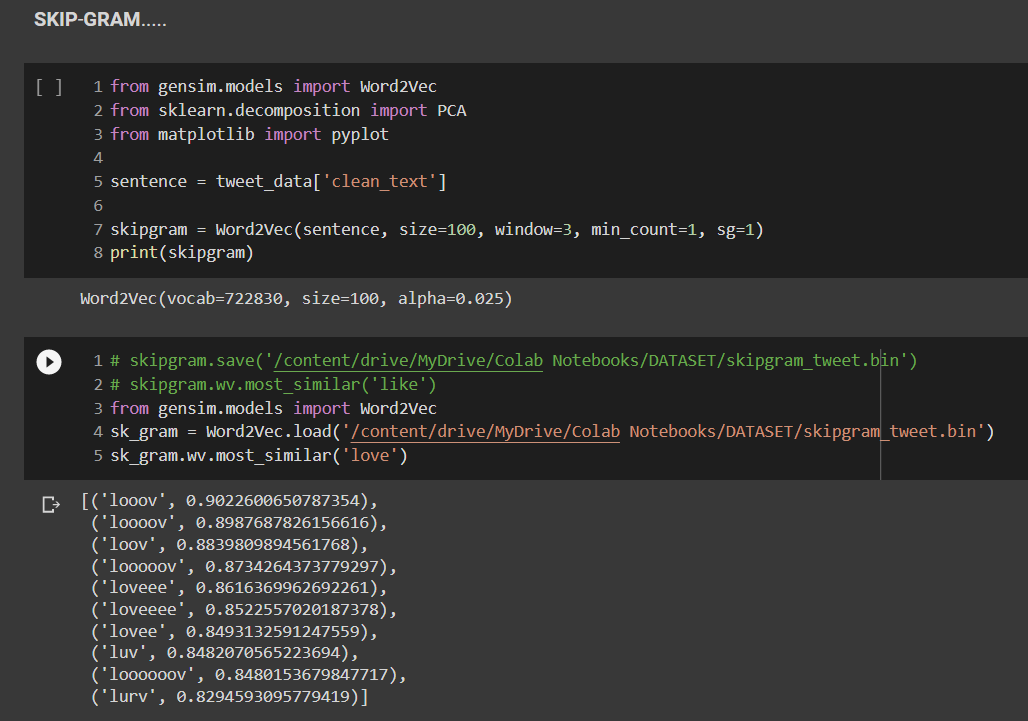


**Figure 3. Bar Chart Visualization of Target (EDA)**

With the equal bars its graphically show that both class (‘positive’ and ‘negative) contain equal numbers of datapoints. However, the ‘id’ columns denote the identity code for each user on the tweeter platform and ‘data’ columns keep track of tweet timestamp (day, month, hour, minute, seconds and year), ‘user’ columns contain username and finally the ‘text’ column holding the tweet of every user. Its identity that certain columns on the raw dataset have no contribution for the sentiment classification problem, while the actual dataset consist of symbol, stop-words, and slag words. Hence, its essential to carry out data preprocessing and cleaning.

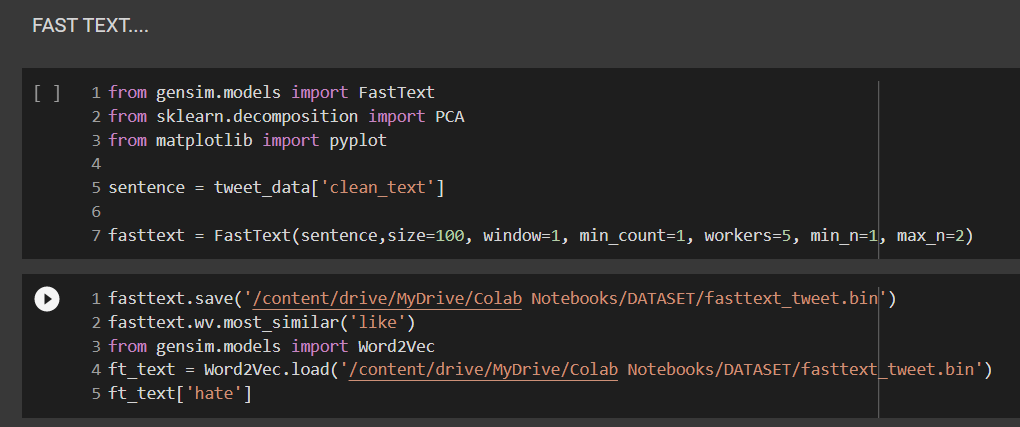
**4.3 Feature Extraction (Skip-Gram, Fast-Text)**

In the field of Natural Language processing feature extraction process entail extracting important information or description of words in a vocabulary. This sturdy adopts the FastText and Skip-gram approach to extraction feature vector of words in the vocabulary. The figure 4 and 5 below shows how the Skip-gram and Fasttext word embedding feature vector are achieved



**Figure 4. Word Embedding (Skip-gram implementation)**

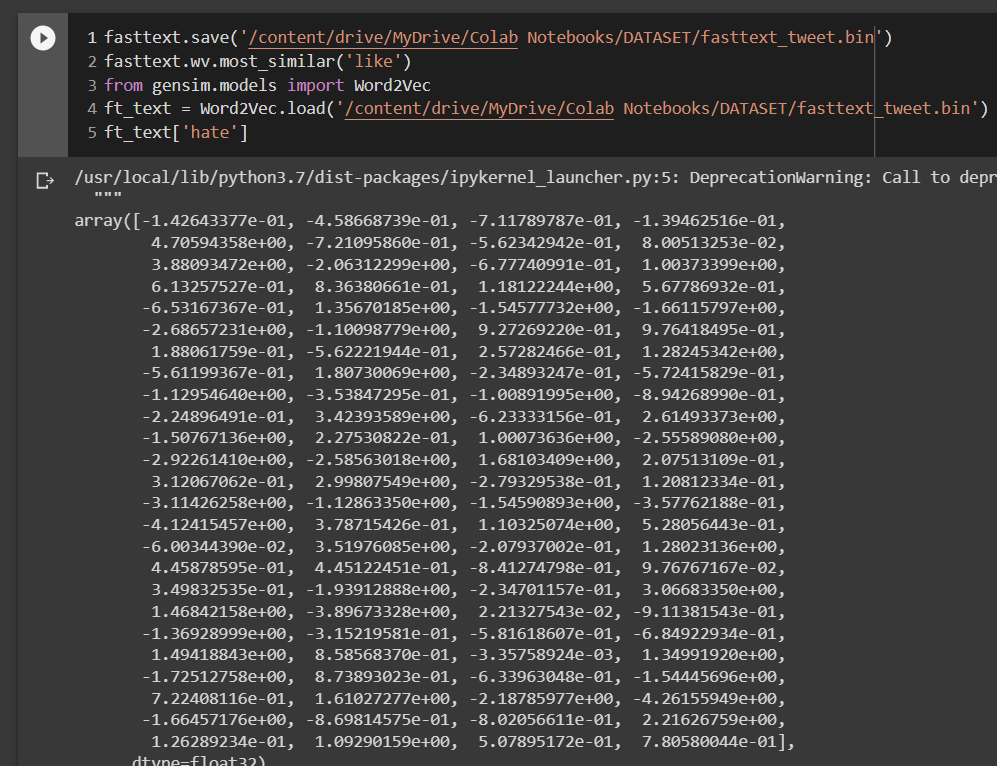
Based on the figure 4, the word embedding is implemented using the genism Word2vec module. The raw dataset is passed as parameter along with the word embedding vector size. Based on the figure about 722830 vocabularies is generated and each word as a vector size of 100. The resulted weighted features extracted can be save and loaded at any time using the save and load helper method. Finally, the word2vec is tested in this instance by computing the similarity score of the word ‘love’.



\

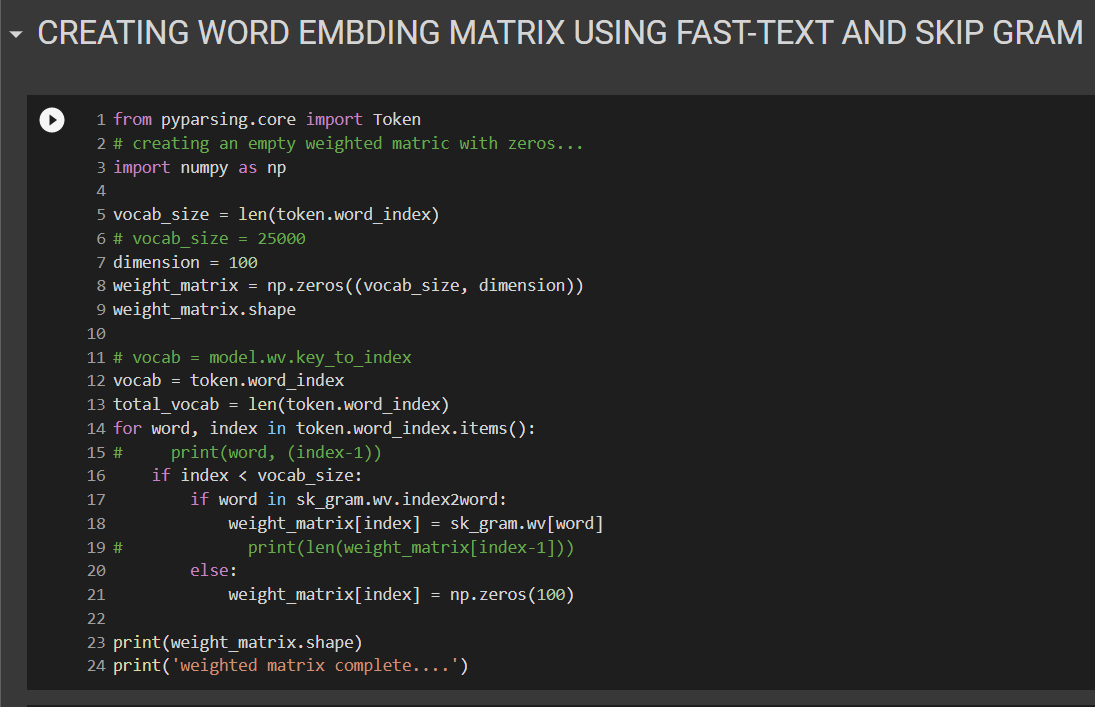
**Figure 5. Word Embedding (FastText implementation)**

The fastText also adopt the same method as Skip-gram with slit variation on the method of extracting of features. the same genism module provides and API to access FastText functionalities. In this instance 100-dimensional vector of each vocabulary is generated with window size of 1. The figure 6a shows how the word embedding looks likes numerically, for the word ‘hate’ it’s visualized in the figure below.



**Figure 6a. Word Embedding (FastText) for “hate”**

Moreover, to feed the feature vector of Skip-gram and FastText to the deep neural network embedding layer its required to generated a weighted matrix (matrix of feature vector) holding the weights or feature vector of each word in the genism vocabulary. The weighted matrix is actualized with the code fragment in the figure 6b below, and the dimension of the matrix is equivalent to the size of the vocabulary cross 100 for each words.

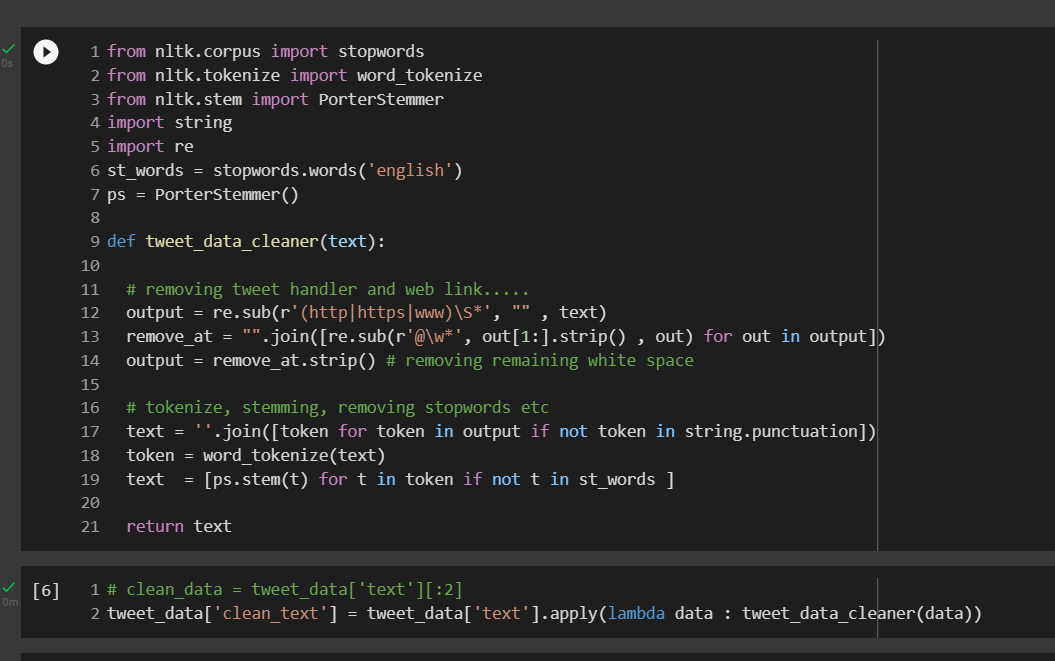


**Figure 6b. Word Embedding (FastText) for “hate”**

**4.4 Data Preparation and Cleaning**

this section includes preparation of dataset for deep learning understanding and training, it also involves the cleaning processes such as

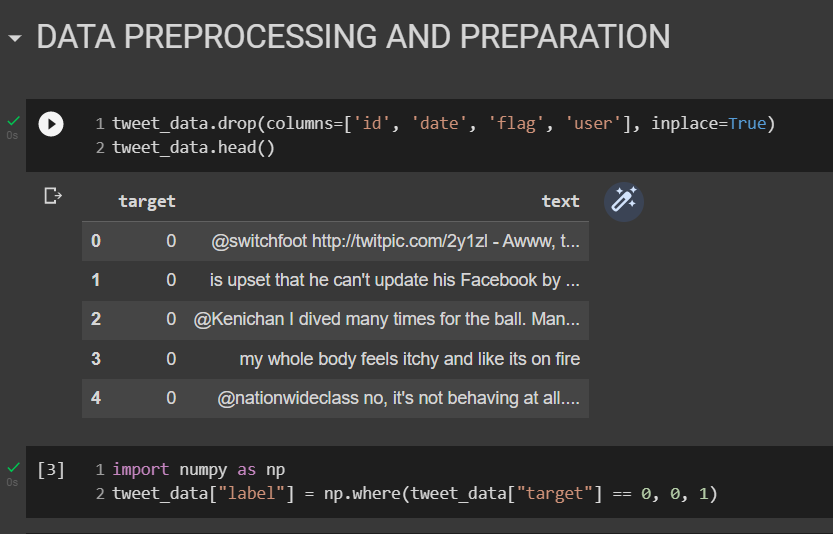
1. Stemming (Porter Stemming algorithm) : the stemming involve minimizing all words in the text corpus to their stem or root words
2. Removal of punctuation such as (@, #$%^&\*() {}><?”, etc.)
3. Removal of links: the text corpus contains various links such as email, and web links.
4. Removal of user handler: Every tweet is denoted using user handlers which start with the ‘@’ symbol and handler name
5. Removal of Stop-Words: stop words are word found in every datapoint that has no significant meaning to the text corpus.



**Figure 7. Tweet Text preprocessing**

The figure7 shows the code fragment used in cleaning the tweet corpus data point using the Natural Language Processing Toolkit (NLTK) functionality such as stemming, word tokenization, stop words and the likes.

Furthermore, certain preparation has to be carry out such as removing unnecessary attribute or feature (columns) from the main dataset. The ‘Id’, ‘date’, ‘flag’, and the ‘user’ columns are remove since it has no significant contribution to the classification problem. However, the ‘target’ and the ‘text’ columns are extracted and a new DataFrame (tabular representation of the raw dataset) is formulated. The figure 8 below shows how the new DataFrame is formulated.



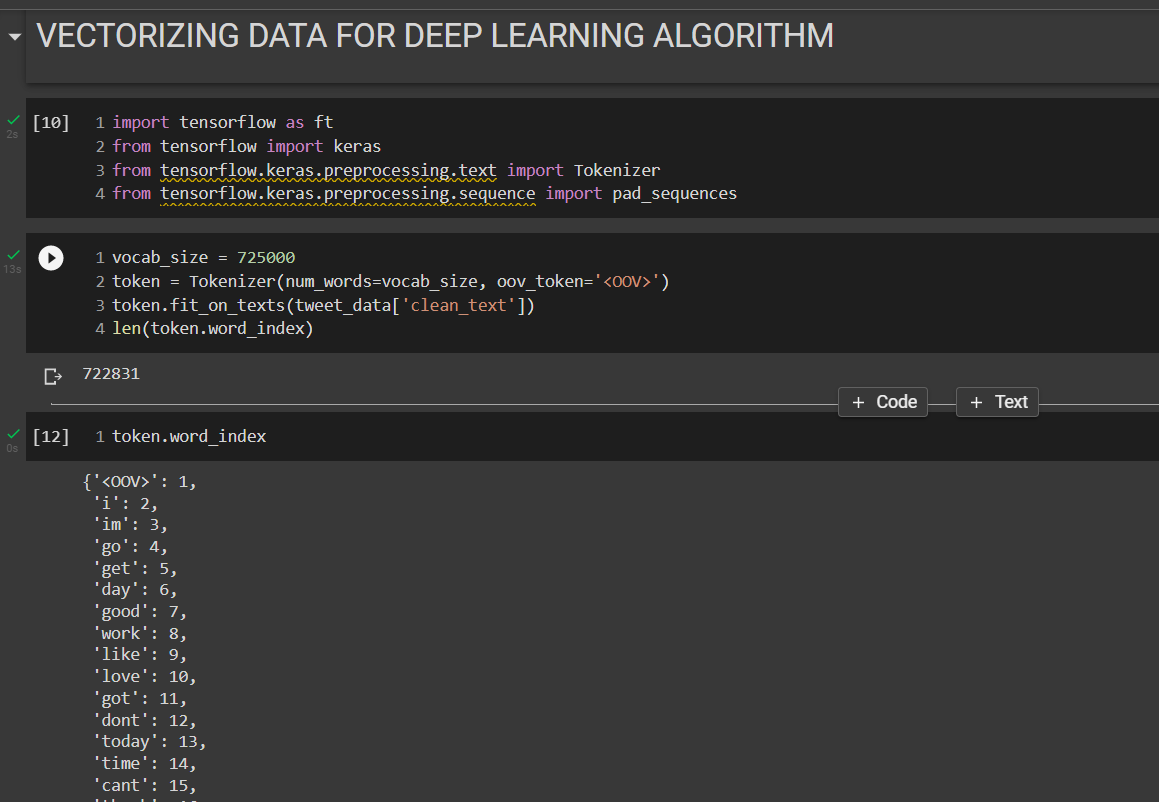
**Figure 8. New Dataframe by removing irrelevant feature (column)**

The **figure 8** also shows how the label with (‘0’ = negative and ‘4’ = positive) is changed or reassigned to (‘0’= negative and ‘1’= positive) using the numpy application programming interface (API). This process ensure that the text and label are cleaned for efficient processing. However, further preprocessing that is need will be discuss in the next sub-section.

**4.5 Further Preprocessing (Vectorization)**

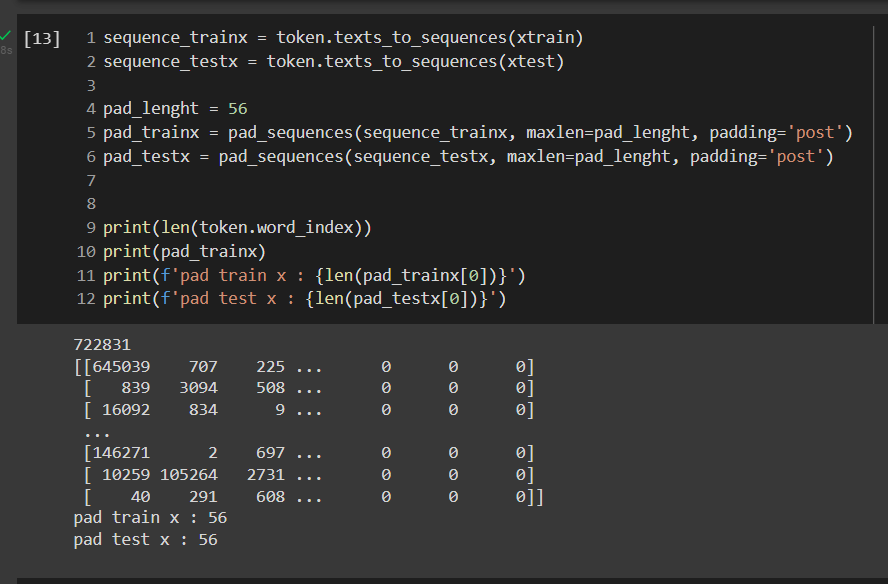
The further preprocessing considered in this study is essential for easy processing by the hybrid-deep learning model thus, the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU). This further preprocessing includes

1. **TensorFlow** **Tokenization**: splitting text document into is component words.
2. **Building** **the** **Vocabulary**: this approach uses a dictionary like data structure to associate each word in a text corpus to a unique number.
3. **Sequencing**: this uses One-Hot encoding approach to convert text to it unique id (number) in the vocabulary
4. **Padding**: Since each datapoint in the tweet dataset in most cases are of different length, its essential to pad all data point with zero’s just to have equal lent of document.



**Figure 9. Tokenizer, vocal building and sequencing**

The figure nine shows how the tokenizing operation is perform using the TensorFlow api and the parameters that are passed in. based on this sturdy a vocabulary of 725,000 is generated and the tweet text are convert to numbers using does 725,000 unique words. The same vocab is printed out in the figure nine as follows (‘i’:2 , ‘in’:3 etc). however, figure ten show the padding operation using the TensorFlow pad sequences API, and each document in the tweet dataset are padded with certain number of zero’s to give the perception of equal length words.



**Figure 9. padding operation using pad sequence.**

**4.4.1 Data Splitting**

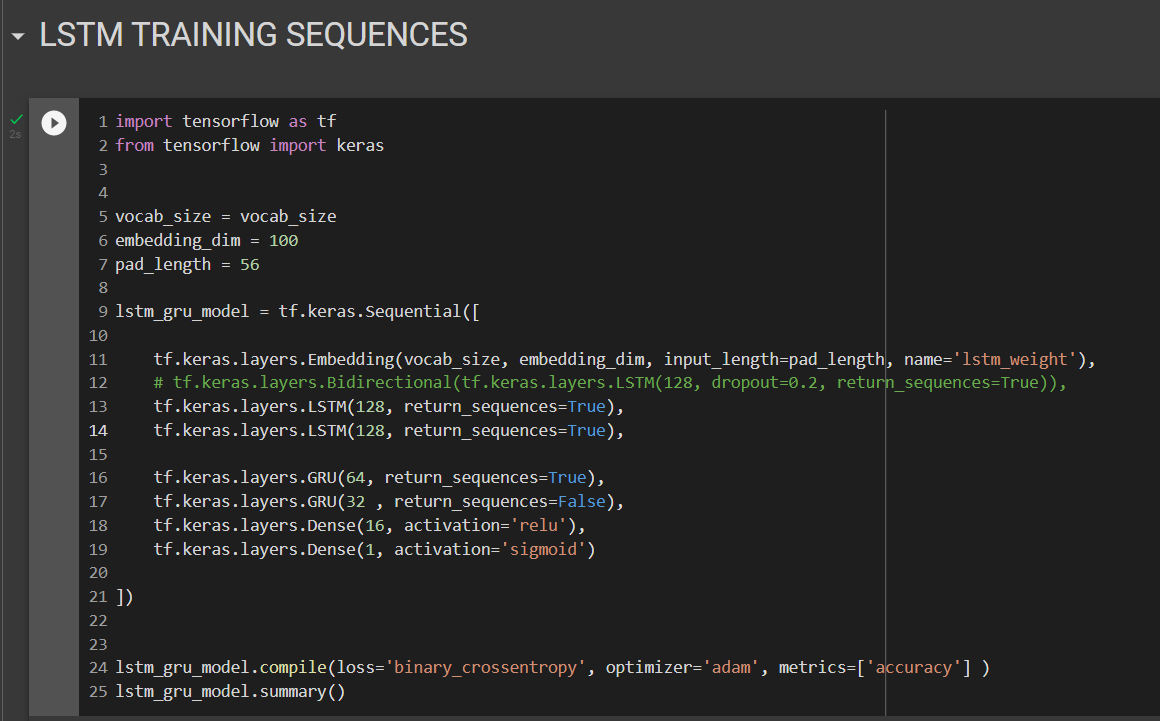
The data splitting involves slicing of main dataset into training (usually much) dataset and testing (smaller in comparison to training set). Based on the thesis that dataset is split into 80 percent training set and 20 percent testing set, which resulted into 1,280,000 million training datapoint and 230,000 thousand testing datapoint. The figure ten shows the code sample for achieving data splitting.



**Figure 10. sentiment 140 data splitting.**

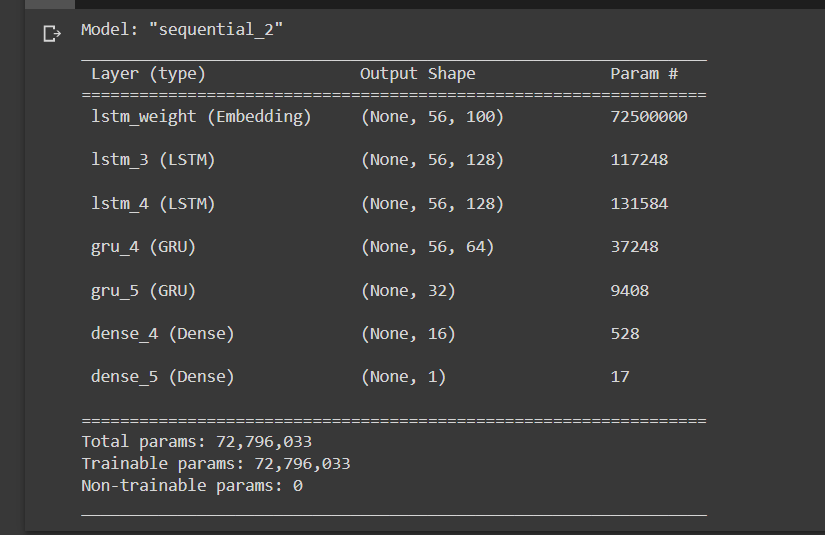
**4.5 LSTM-GRU Model (Network Configuration)**

The proposed model hybridized the Long short-term memory and Gated Recurrent Unit network for the training of the tweet sentiment140 dataset. Considering the figure 11 showing the code snippet for defining the network stack (input, embedding, LSTM, GRU, dense and output layer).



**Figure 11. LSTM-GRU Model**

Furthermore, the vectorization vocab size is assigned to the vocabulary size, embedding dimension is set to 100 and a padding length as 56. The LSTM-GRU model is define using the sequential API with one embedding, two layers of LSTM, two layers of GRU, one dense and output layer. The embedding layer accept vocabulary size, input length, embedding dimension as parameter, each LSTM contain 128 neurons with return sequence to be true. However, the model also contains two layers of Gated Recurrent Unit, the first GRU contain 64 neurons and the second layer contain 16 neurons with ‘relu’ as the activation function for both GRU layers. Finally, the model is compile with a loss function set to ‘binary\_crossentropy;, optimizer set to ‘adam’, and the performance metric is set to ‘accuracy. The model configuration is summarize using the summary helper method and its show in figure 12 below.



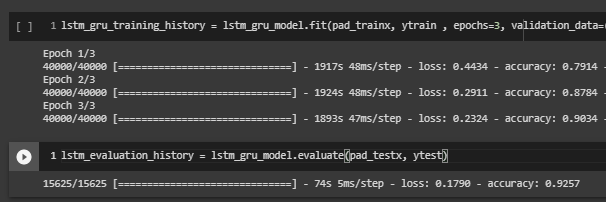
**Figure 12. LSTM-GRU Model Summary**

the figure 12, show a summary configuration of the model by specifying each layer, output shape and param as column. A total parameter of 72,796,033 is generated with 72,796,033 trainable and 0 non-trainable parameter.

However, a word embedding matrix is generated to hold or assign the weight of each word embedding representation in the vocabulary. A weighted matrix of dimension vocab size by 100 if generated. Then finally the weighted matrix is passed to the proposed hybrid model via the embedding layer.

**4.6 Result and Discussion**

The Proposed Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) model is trained with the training datapoint in three (3) epoch (Iterations) with training accuracy of 0.9034 and testing accuracy of 0.9257 percent is achieved



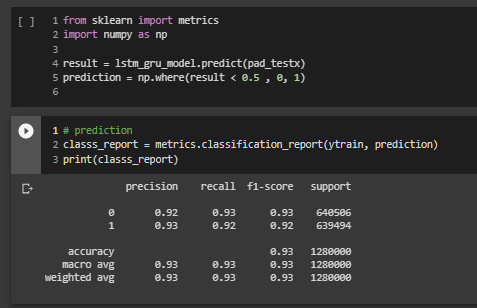
**Figure 13. LSTM-GRU training and evaluation process**

Moreover, to access the performance of the proposed model various performance metrics is considered such as;

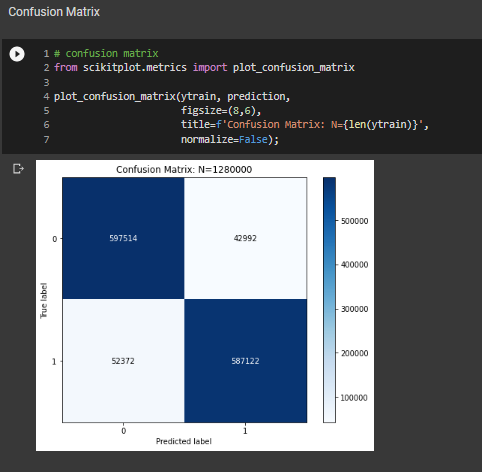
1. **Accuracy**: this denote an approach used in measuring the performance of machine or deep learning model. It basically expresses in percentage and it calculate the prediction value against the true prediction value.
2. **Precision**: This is one of the machine learning model performance indicator, which shows the quality of a positive prediction that is made by the developed model.
3. **Recall**: This metrics compute the ratio between the number of positive samples that are correctly classify against the positive total number of positive samples.
4. **F1**-**score**: It’s the measure of model’s accuracy on each classes in a dataset.

The figure 14 show the result gotten under the performance metrics listed earlier, using the inbuilt ‘sklearn’ python module. Based on the figure the accuracy of 92% is achieved for class negative sentiment, while 93% percent is achieved for the positive sentiment class and both class achieved and average accuracy of 93%.

Considering figure 15 show the performance or the developed mode using the confusion matrix, the confusion matrix shows the



**Figure 14. LSTM-GRU Model Classification Report**



**Figure 15. LSTM-GRU Model Confusion Matrix Report**

the confusion matrix also known as the error matrix, this shows the tabular visualization of the model performance. The two-dimensional table show the actual value of each class over the predicted value. In this case the ‘0’ or negative tweet sentiment true predicted value is shown against the true and false predicted value, and the ‘1’ positive tweet sentiment true predicted value is compare against the true and false predicted value.

**Table 4.1: Performance comparison with existing model**