Department of Decision Science

Faculty of Business

University of Moratuwa

Semester 7

DA4210 – Text Analytics

Assignment 01

Fraud News Detection using Text Analytics

Name: M. W. S. A. U. SILVA

Index No: 206121X

Due Date of Submission

[08 / 05 / 2024]

Contents

1. Problem Selection

- **Topic:** Fake News Detection using Text Analytics

- **Overview:** The integrity of information spread is seriously challenged by the spread of fake news in today's digital environment. The quick dissemination of false information via social media, news websites, and online forums has made it more difficult to separate between true and fake news stories. In addition to misleading the public, fake news endangers political stability, social cohesiveness, and public confidence in the media. Therefore, it is essential to create strong methods for identifying and reducing the impact of fake news.

- **Significance:** The spread of false information can have serious consequences for people, groups, and even entire countries. False information on social events, politics, the economy, and public health can cause people to make poor judgments, increase social unrest, and threaten democratic processes. Thus, to protect the integrity of information ecosystems and encourage well-informed decision-making, it is imperative to put into practice efficient methods for spotting and neutralizing fake news.

- **Relevance of Text Analytics:** By using natural language processing (NLP) techniques to efficiently evaluate the content of texts, text analytics plays a crucial role in addressing the problem of false news identification. Through the use of linguistic analysis and machine learning algorithms, text analytics makes it possible to extract valuable insights from massive amounts of unstructured text data. From these insights, patterns can be found, linguistic indicators suggestive of false news can be found, and predictive models for automated detection can be created.

Moreover, text analytics makes it easier to integrate different data sources, such as user interactions, contextual data, metadata, and textual content, to improve the precision and dependability of fake news detection systems. Through the utilization of text analytics, we can create new methods for recognizing deceptive language patterns, evaluating the reliability of sources, and identifying irregularities suggestive of disinformation efforts.

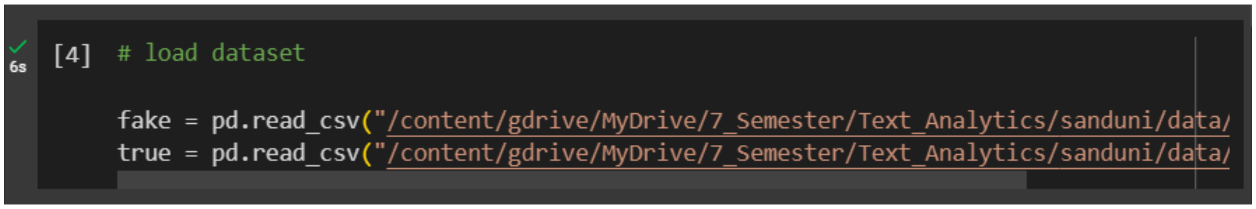
2. Data Collection

- **Source of the dataset:** The dataset, which concentrated on subjects related to text analytics and natural language processing (NLP), was taken from Kaggle's collection of openly available datasets.

- **Characteristics of the dataset:** The collection is built up of text that has been taken out of articles and covers a wide range of subjects, authors, and writing styles. It has two separate files as true.csv and false.csv.

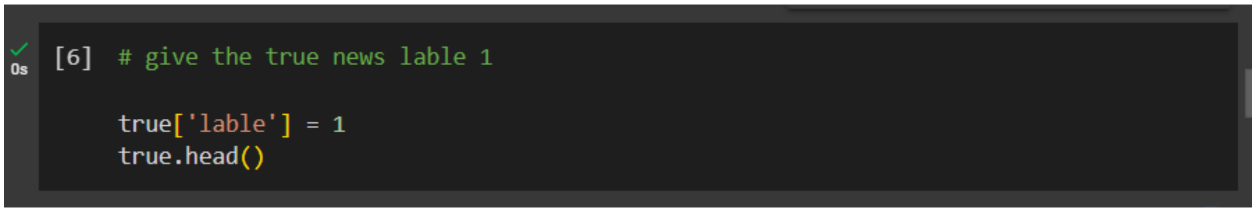
3. Preprocessing

Step 1: load dataset

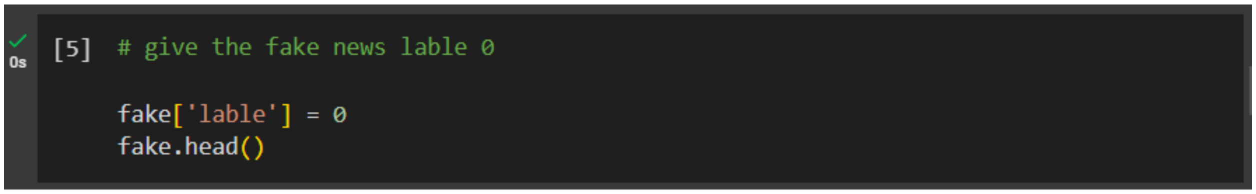


```
[4]  # load dataset

     fake = pd.read_csv("/content/gdrive/MyDrive/7_Semester/Text_Analytics/sanduni/data/
     true = pd.read_csv("/content/gdrive/MyDrive/7_Semester/Text_Analytics/sanduni/data/
```

Step 2: label true news as 1 and false news as 0

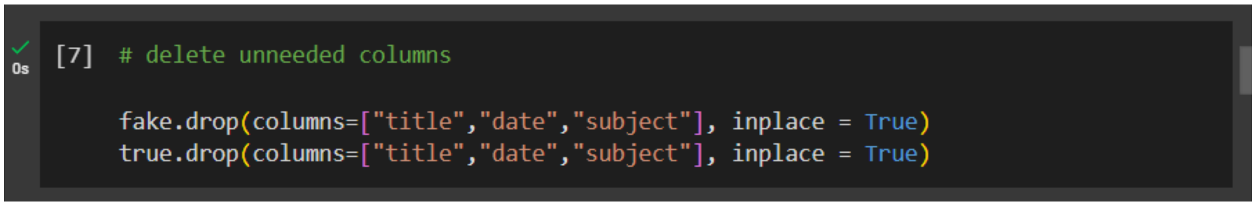


```
[6]  # give the true news lable 1

     true['lable'] = 1
     true.head()
```



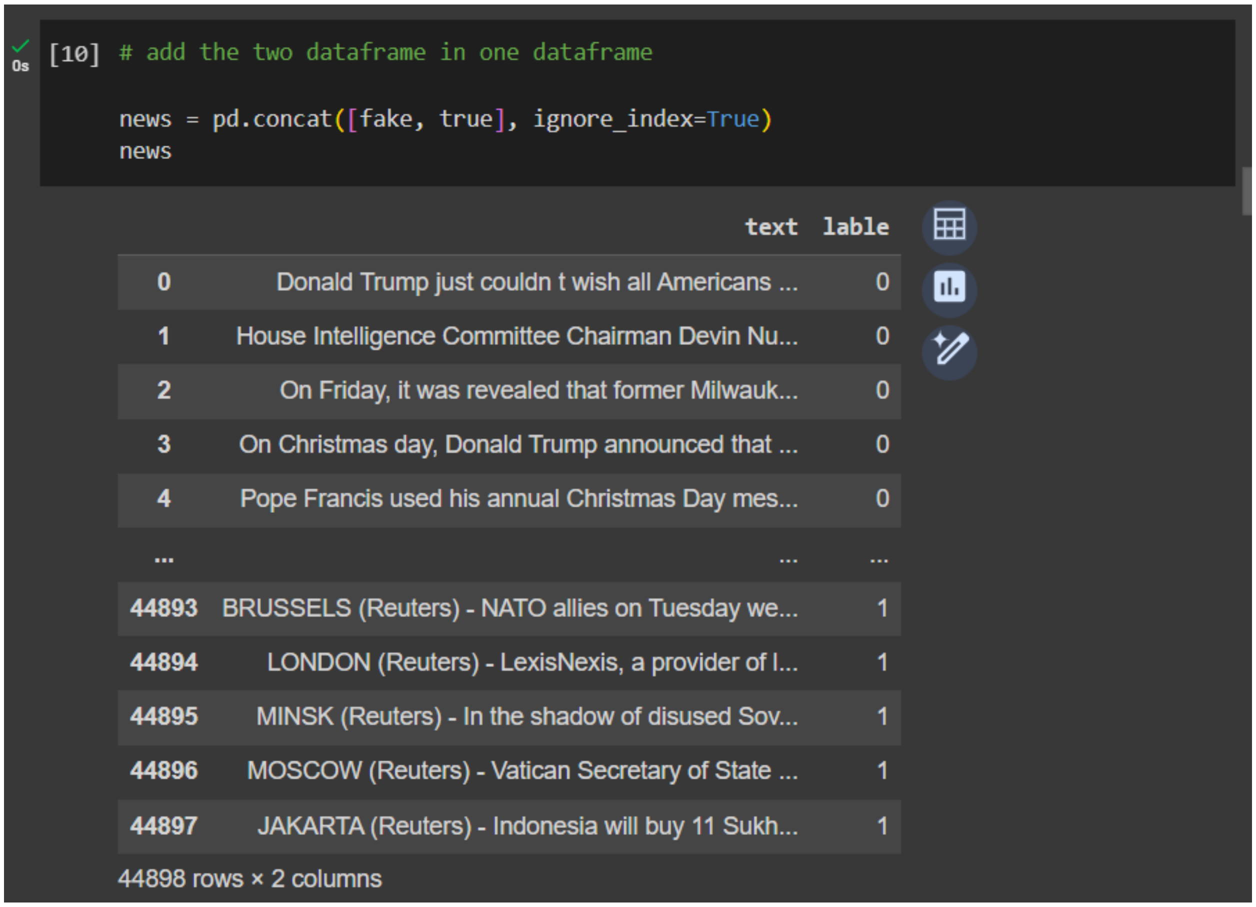
```
[5]  # give the fake news lable 0

     fake['lable'] = 0
     fake.head()
```

Step 3: delete unnecessary columns (title, data, subject)



```
[7] # delete unneeded columns

    fake.drop(columns=["title","date","subject"], inplace = True)
    true.drop(columns=["title","date","subject"], inplace = True)
```

Step 4: connect two datasets to one dataset

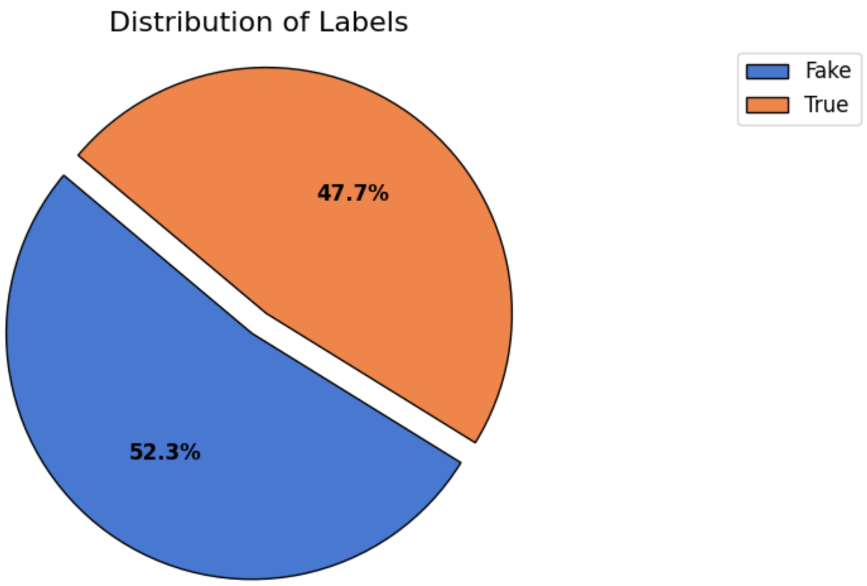


```
[10] # add the two dataframe in one dataframe

     news = pd.concat([fake, true], ignore_index=True)
     news
```

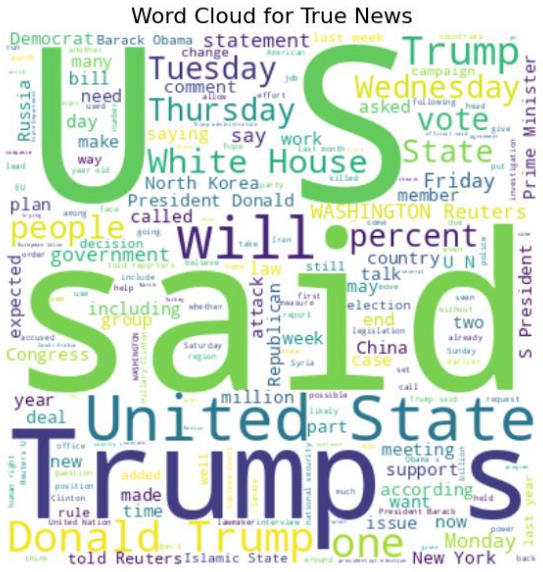
```
                                                  text  lable
0        Donald Trump just couldn t wish all Americans ...      0
1        House Intelligence Committee Chairman Devin Nu...      0
2        On Friday, it was revealed that former Milwauk...      0
3        On Christmas day, Donald Trump announced that ...      0
4        Pope Francis used his annual Christmas Day mes...      0
...                                                    ...    ...
44893  BRUSSELS (Reuters) - NATO allies on Tuesday we...      1
44894     LONDON (Reuters) - LexisNexis, a provider of l...      1
44895    MINSK (Reuters) - In the shadow of disused Sov...      1
44896    MOSCOW (Reuters) - Vatican Secretary of State ...      1
44897    JAKARTA (Reuters) - Indonesia will buy 11 Sukh...      1
44898 rows × 2 columns
```

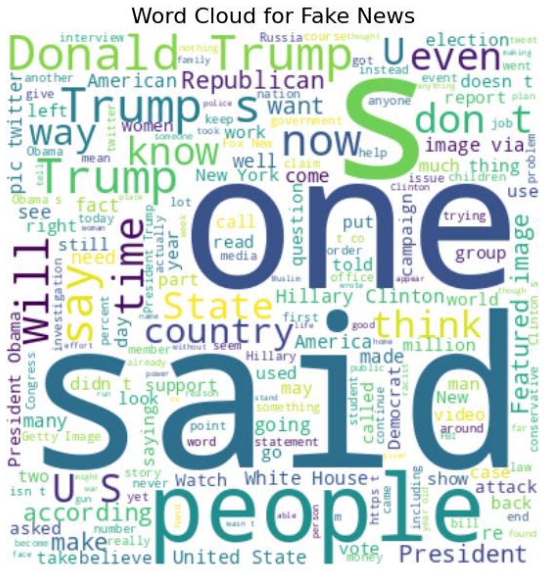
Step 5: Exploratory Visualizations

- Label Distribution

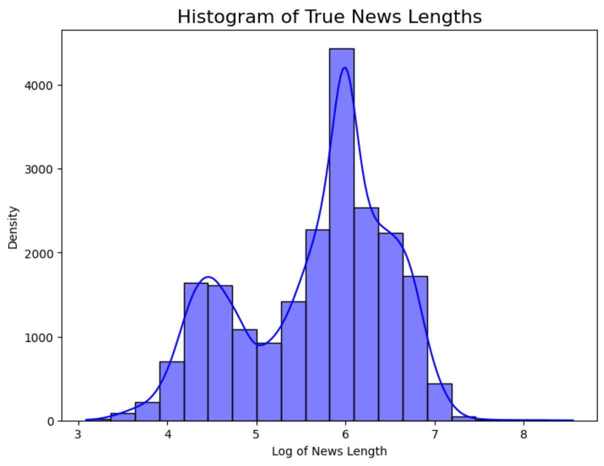


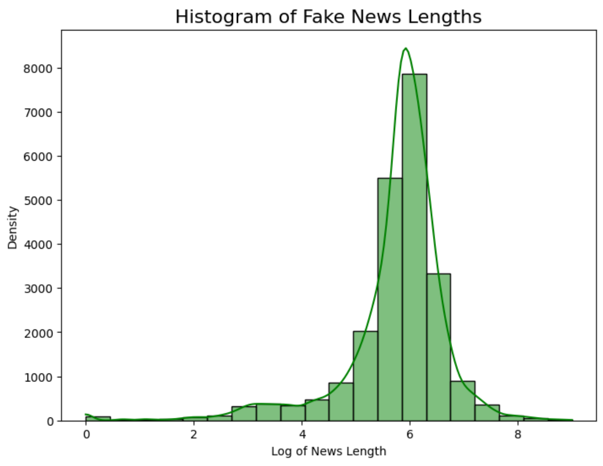
- Wordcloud



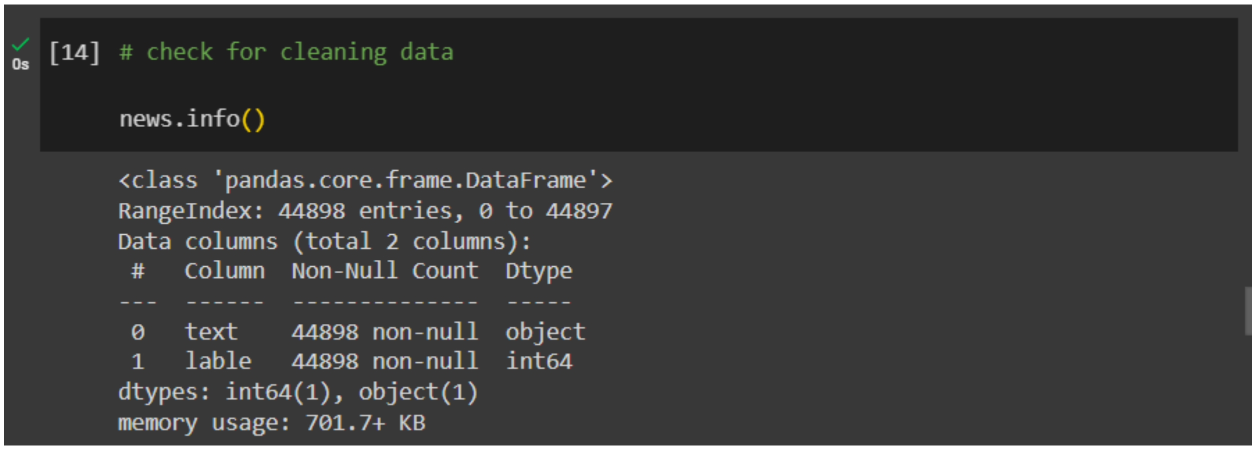


- News Length Distribution





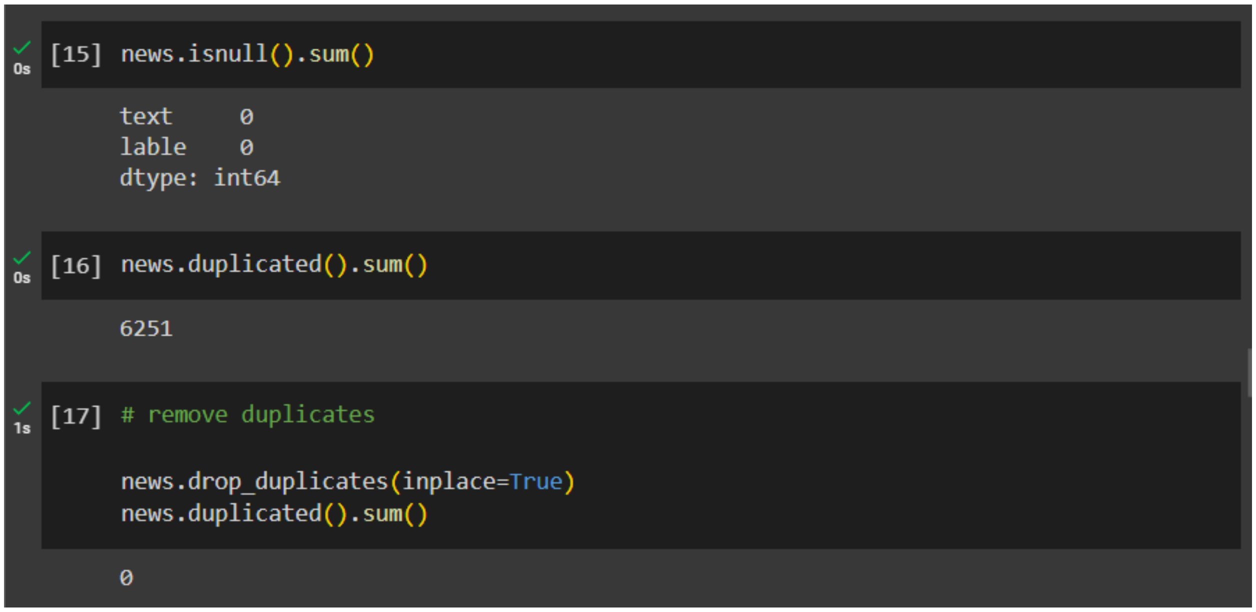
Step 6: Check for cleaning data



```
[14] # check for cleaning data

     news.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44898 entries, 0 to 44897
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    44898 non-null  object
 1   lable   44898 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 701.7+ KB
```

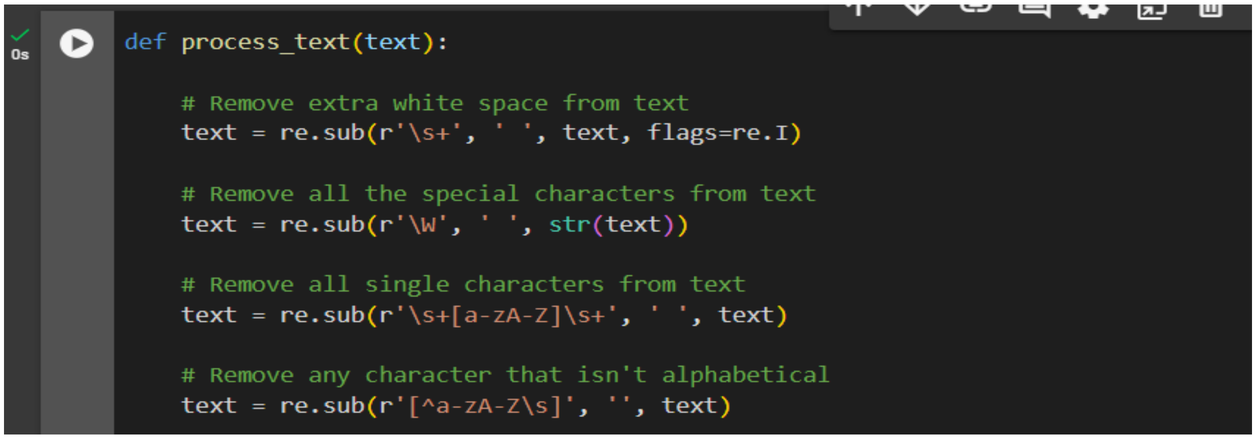
Step 7: since no null values, in this step duplicates were removed



```
[15] news.isnull().sum()

     text     0
     lable    0
     dtype: int64

[16] news.duplicated().sum()

     6251

[17] # remove duplicates

     news.drop_duplicates(inplace=True)
     news.duplicated().sum()

     0
```

Step 8: in this step, all the other main preprocessing steps are done at once.

- Remove extra white spaces, remove all the special characters, remove all single characters, remove all non-alphabetical characters:
  - ➢ By eliminating characters and symbols that are unnecessary and do not add to the text's semantic meaning, these procedures help in the cleaning of the text data. Eliminating single and special characters facilitates the focus on meaningful words and increases the effectiveness of subsequent processing processes. Removing unnecessary white spaces promotes uniformity.



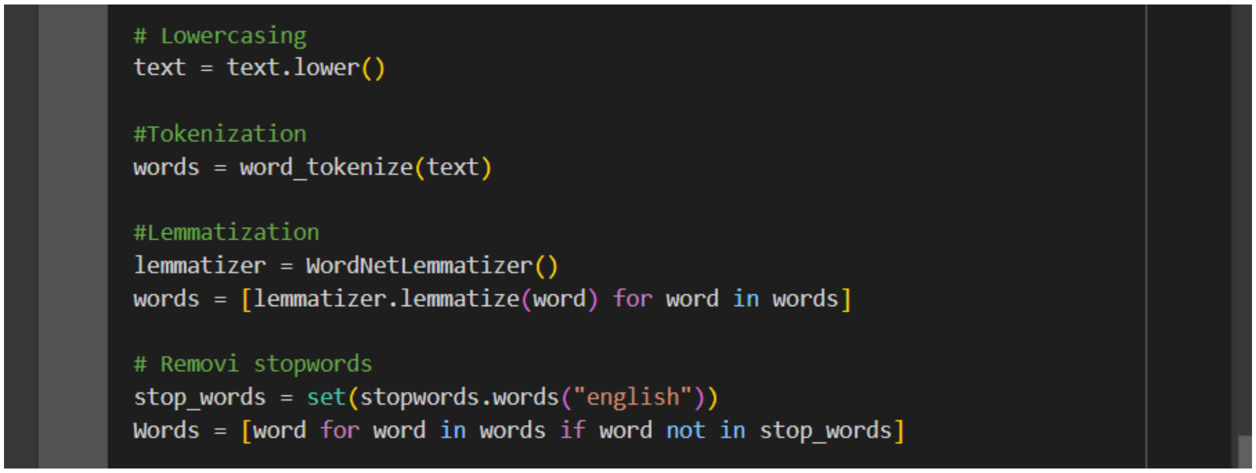
```
def process_text(text):

    # Remove extra white space from text
    text = re.sub(r'\s+', ' ', text, flags=re.I)

    # Remove all the special characters from text
    text = re.sub(r'\W', ' ', str(text))

    # Remove all single characters from text
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)

    # Remove any character that isn't alphabetical
    text = re.sub(r'[^a-zA-Z\s]', '', text)
```

- Lowercasing
  - ➢ As lowercase and uppercase forms of a word are treated equally, lowercasing the entire text guarantees uniformity in word representations. By avoiding feature duplication and reducing the vocabulary, this stage streamlines the text analysis procedure.

- Tokenization
  - ➢ Tokenization separates the text into individual words, or tokens, to make it easier to extract features and conduct additional analysis. Tokenization allows the use of different NLP techniques, such as lemmatization and stopword removal at the word level, by dividing the text into meaningful units.

- Lemmatization
  - ➢ Lemmatization, which reduces words to their most basic or dictionary form (lemma), helps in vocabulary normalization and boosts the precision of text analysis tasks. Lemmatization improves the interpretability of results by reducing vocabulary sparsity by returning inflected terms to their base forms.

- Remove stopwords
  - ➢ Eliminating stopwords helps in focusing attention on the most important words and phrases while lowering noise in the data. By removing unnecessary phrases, this phase can increase the efficacy and efficiency of text analysis systems.

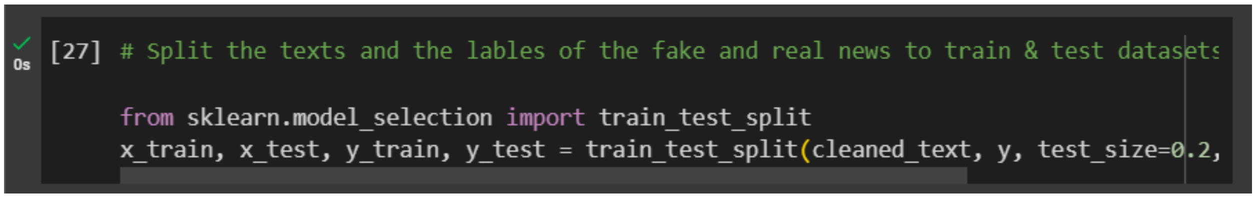


```
# Lowercasing
text = text.lower()

#Tokenization
words = word_tokenize(text)

#Lemmatization
lemmatizer = WordNetLemmatizer()
words = [lemmatizer.lemmatize(word) for word in words]

# Removi stopwords
stop_words = set(stopwords.words("english"))
Words = [word for word in words if word not in stop_words]
```

4. Text Analytics

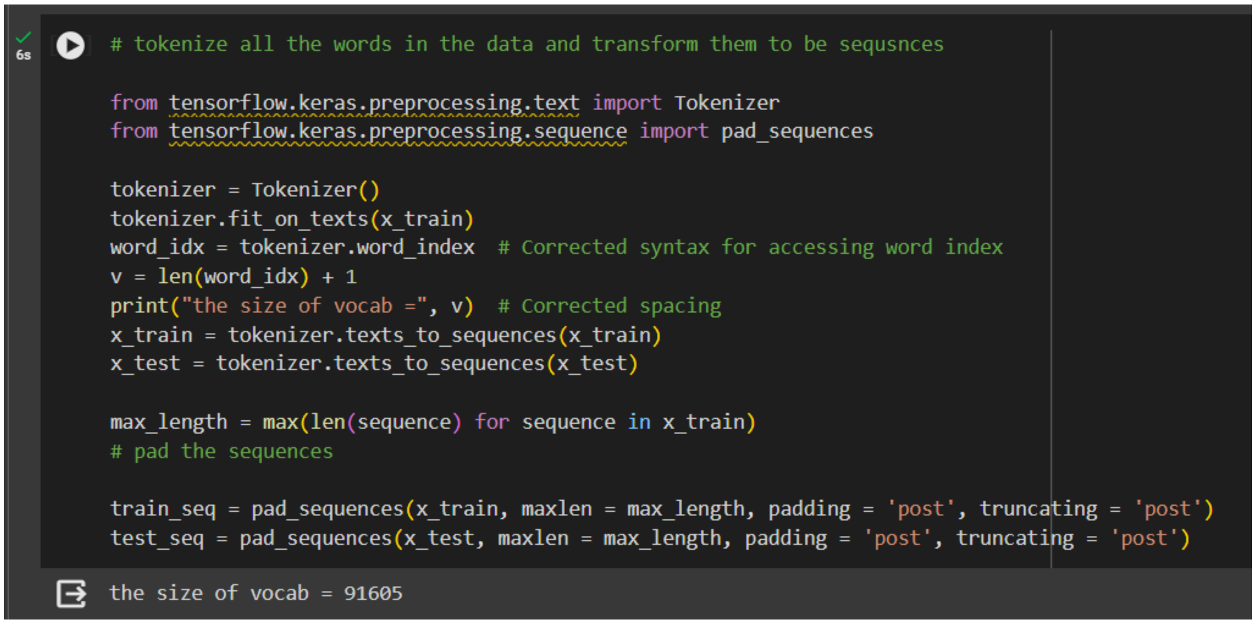
- **Tool:** Text Classification
- **Used Libraries:** nltk, sklearn, tensorflow, etc.
- **Rationale behind Text Classification:** Natural language processing (NLP) uses classification of text extensively to group text items into pre-established groupings or categories. Text categorization makes it possible to automatically classify news stories to be genuine or fraudulent depending on their content when it comes to fraud news detection. Text classification algorithms can learn to recognize language patterns, semantic clues, and characteristics suggestive of bogus news items by training machine learning models on labelled datasets.

Step 1: split the dataset into train and test data where test_size = 0.2



```
[27] # Split the texts and the lables of the fake and real news to train & test datasets

     from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(cleaned_text, y, test_size=0.2,
```

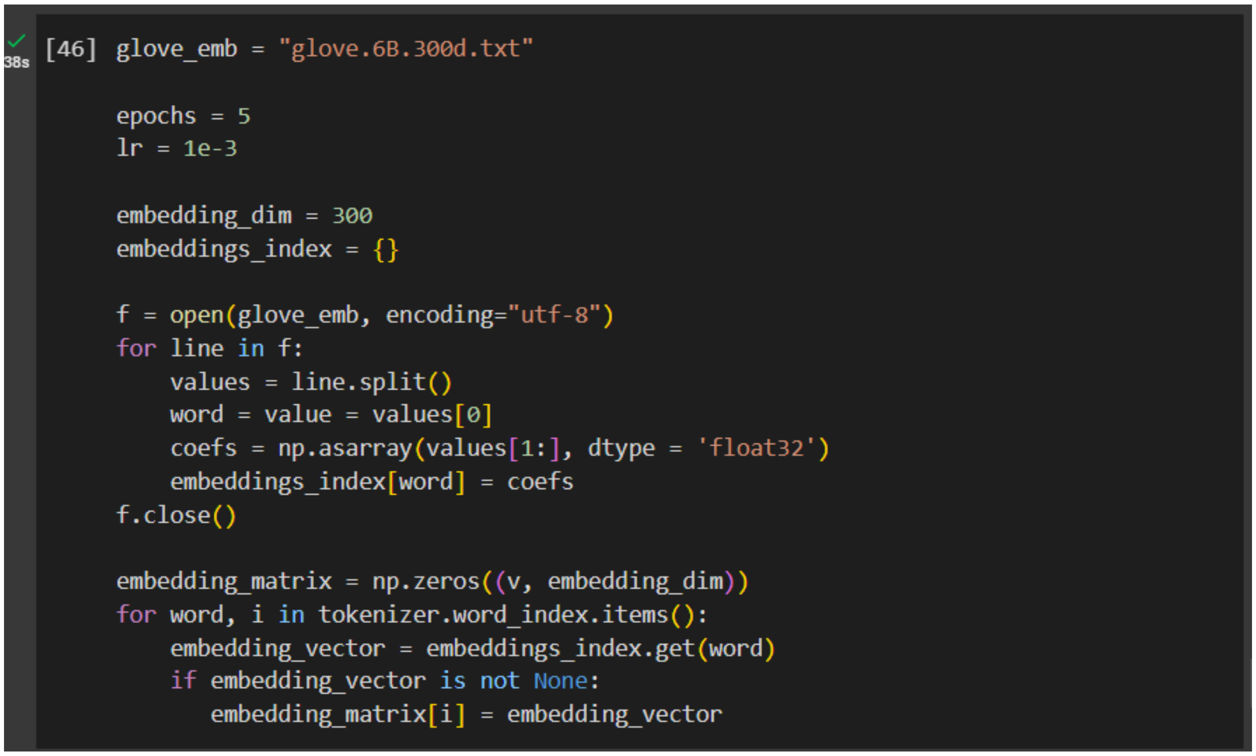
Step 2: tokenize all words and transform them as sequences



```
# tokenize all the words in the data and transform them to be sequsnces

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer()
tokenizer.fit_on_texts(x_train)
word_idx = tokenizer.word_index  # Corrected syntax for accessing word index
v = len(word_idx) + 1
print("the size of vocab =", v)  # Corrected spacing
x_train = tokenizer.texts_to_sequences(x_train)
x_test = tokenizer.texts_to_sequences(x_test)

max_length = max(len(sequence) for sequence in x_train)
# pad the sequences

train_seq = pad_sequences(x_train, maxlen = max_length, padding = 'post', truncating = 'post')
test_seq = pad_sequences(x_test, maxlen = max_length, padding = 'post', truncating = 'post')
```

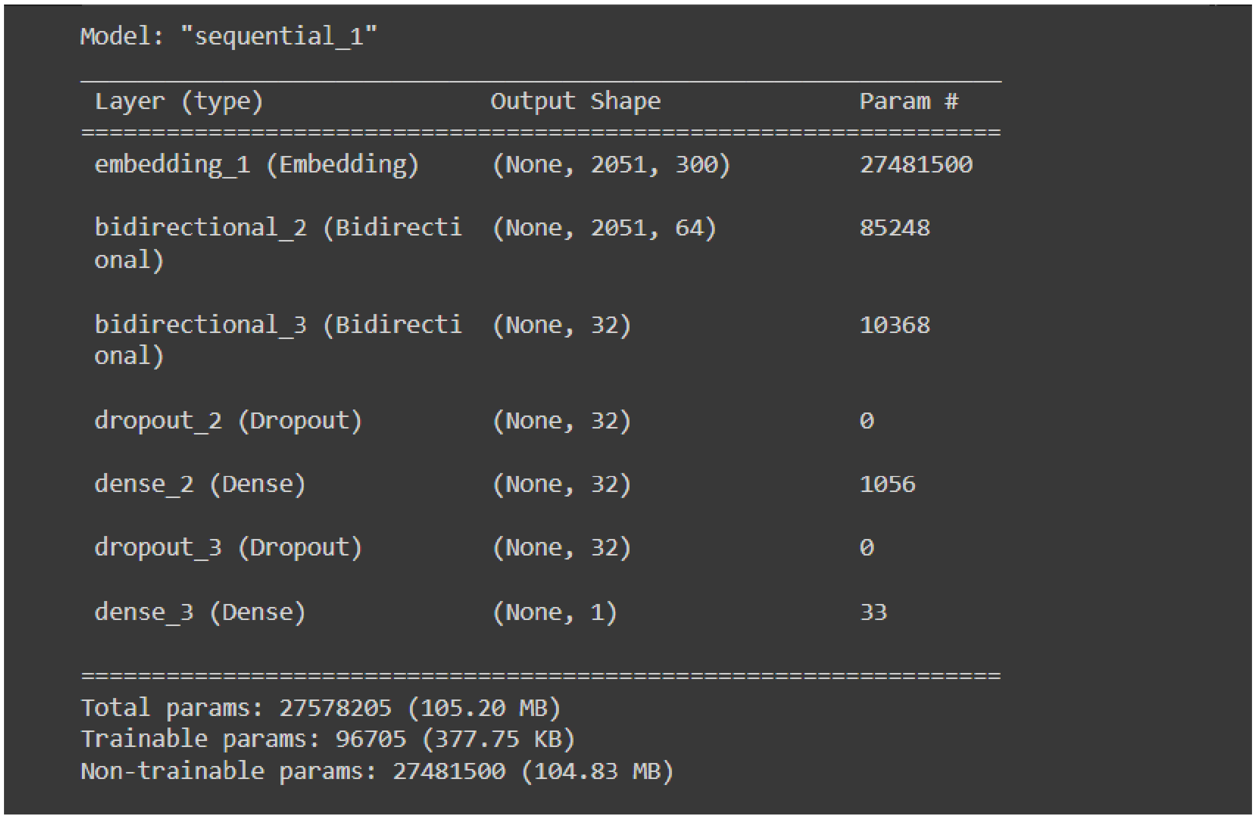
```
the size of vocab = 91605
```

Step 3: Developing classification model



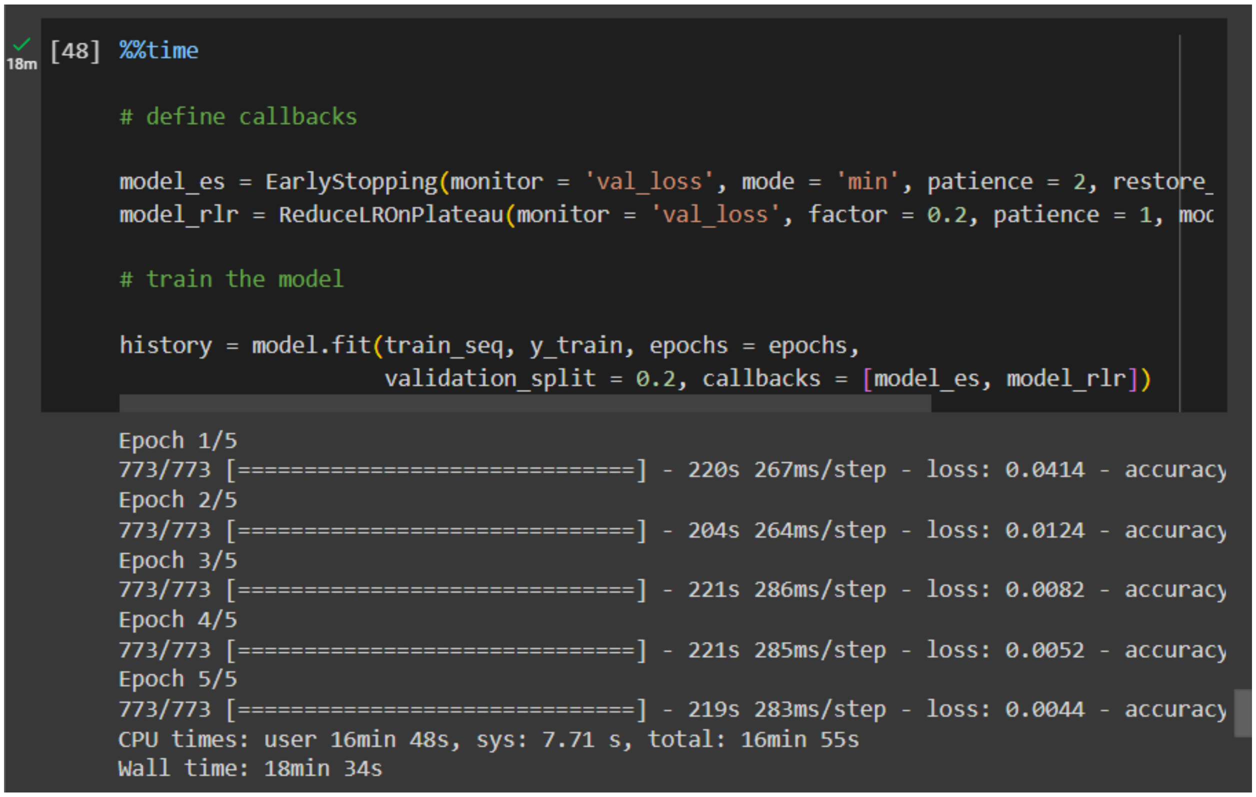
```
[46] glove_emb = "glove.6B.300d.txt"

     epochs = 5
     lr = 1e-3

     embedding_dim = 300
     embeddings_index = {}

     f = open(glove_emb, encoding="utf-8")
     for line in f:
         values = line.split()
         word = value = values[0]
         coefs = np.asarray(values[1:], dtype = 'float32')
         embeddings_index[word] = coefs
     f.close()

     embedding_matrix = np.zeros((v, embedding_dim))
     for word, i in tokenizer.word_index.items():
         embedding_vector = embeddings_index.get(word)
         if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector
```

Step 4: define model



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 2051, 300)         27481500

 bidirectional_2 (Bidirecti  (None, 2051, 64)          85248
 onal)

 bidirectional_3 (Bidirecti  (None, 32)                10368
 onal)

 dropout_2 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 32)                1056

 dropout_3 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 27578205 (105.20 MB)
Trainable params: 96705 (377.75 KB)
Non-trainable params: 27481500 (104.83 MB)
```

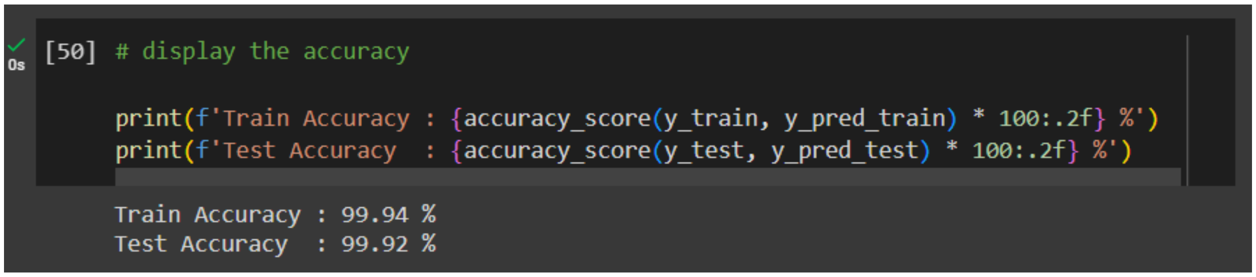
Step 5: train model



```
18m [48] %%time

    # define callbacks

    model_es = EarlyStopping(monitor = 'val_loss', mode = 'min', patience = 2, restore_
    model_rlr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.2, patience = 1, moc

    # train the model

    history = model.fit(train_seq, y_train, epochs = epochs,
                        validation_split = 0.2, callbacks = [model_es, model_rlr])
```

```
Epoch 1/5
773/773 [==============================] - 220s 267ms/step - loss: 0.0414 - accuracy
Epoch 2/5
773/773 [==============================] - 204s 264ms/step - loss: 0.0124 - accuracy
Epoch 3/5
773/773 [==============================] - 221s 286ms/step - loss: 0.0082 - accuracy
Epoch 4/5
773/773 [==============================] - 221s 285ms/step - loss: 0.0052 - accuracy
Epoch 5/5
773/773 [==============================] - 219s 283ms/step - loss: 0.0044 - accuracy
CPU times: user 16min 48s, sys: 7.71 s, total: 16min 55s
Wall time: 18min 34s
```

Accuracy:



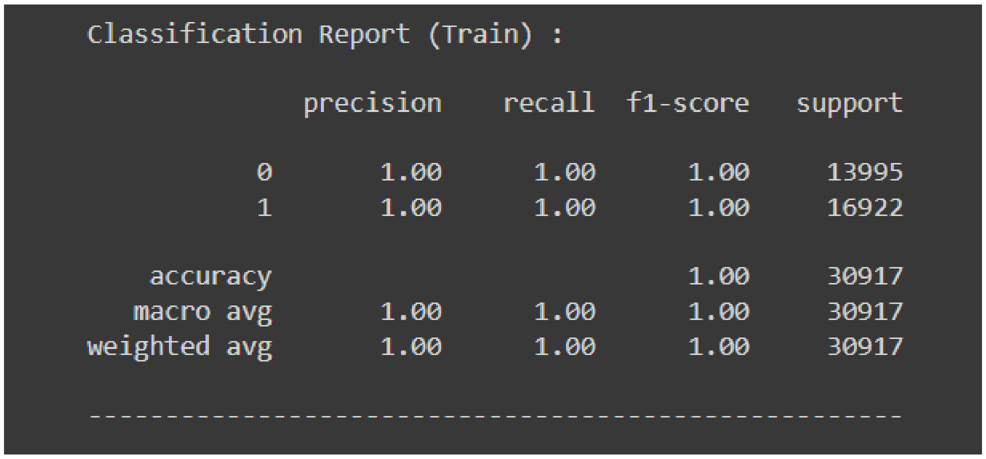
```
[50] # display the accuracy

     print(f'Train Accuracy : {accuracy_score(y_train, y_pred_train) * 100:.2f} %')
     print(f'Test Accuracy  : {accuracy_score(y_test, y_pred_test) * 100:.2f} %')

     Train Accuracy : 99.94 %
     Test Accuracy  : 99.92 %
```

- Train Accuracy: This indicates that on the training dataset, the model's accuracy was a very high 99.94%. Means, 99.94% of the training instances of the model correctly predicted the labels (false or true) when it was trained on a set of labeled examples, in this case news articles. The model appears to have mastered the underlying patterns in the training data, as evidenced by its high train accuracy, which enables it to distinguish between real and fake news articles during training.

- Test Accuracy: The test accuracy of 99.92% suggests that the model did very well on test dataset data, which is unseen data. With an accuracy of 99.92%, the model accurately predicted the labels when tested on a different collection of news stories that it had not seen during training. This high test accuracy indicates that the model is likely to work reliably in real-world circumstances, such as identifying fake news stories in online platforms or media sources, and that it generalizes well to new, unexplored occurrences.

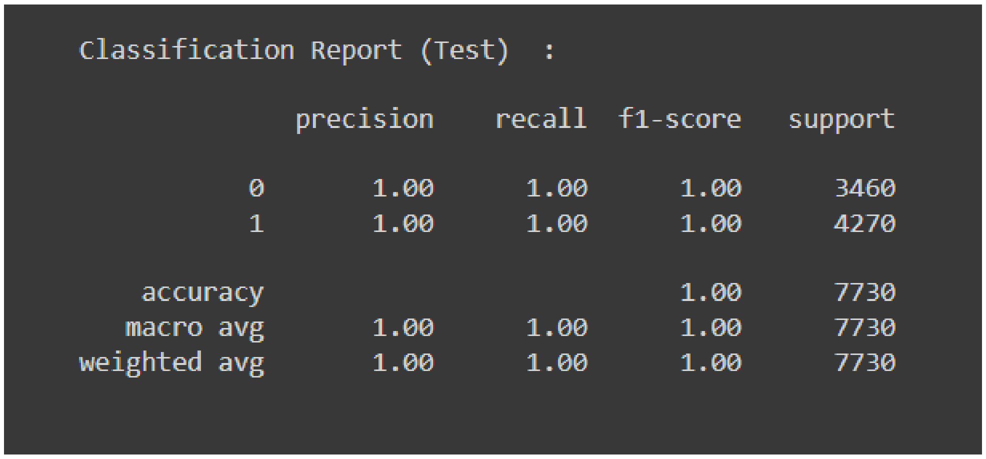
Classification Report:

- Train



```
Classification Report (Train) :

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13995
           1       1.00      1.00      1.00     16922

    accuracy                           1.00     30917
   macro avg       1.00      1.00      1.00     30917
weighted avg       1.00      1.00      1.00     30917

-----------------------------------------------------
```

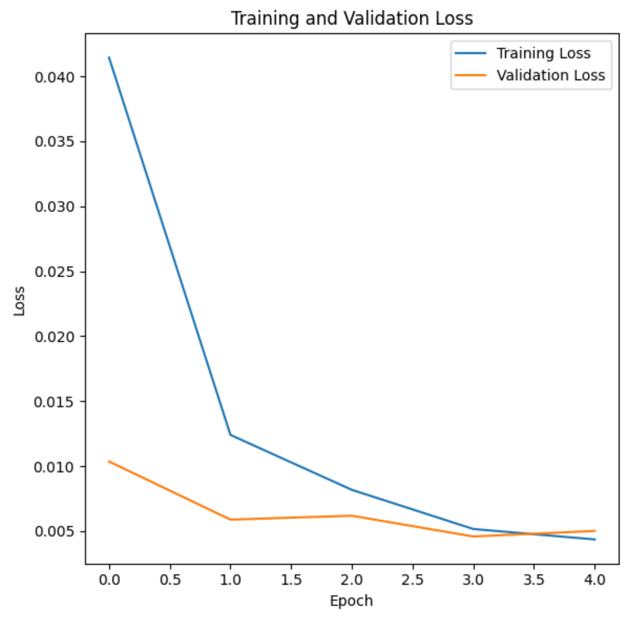
- Test

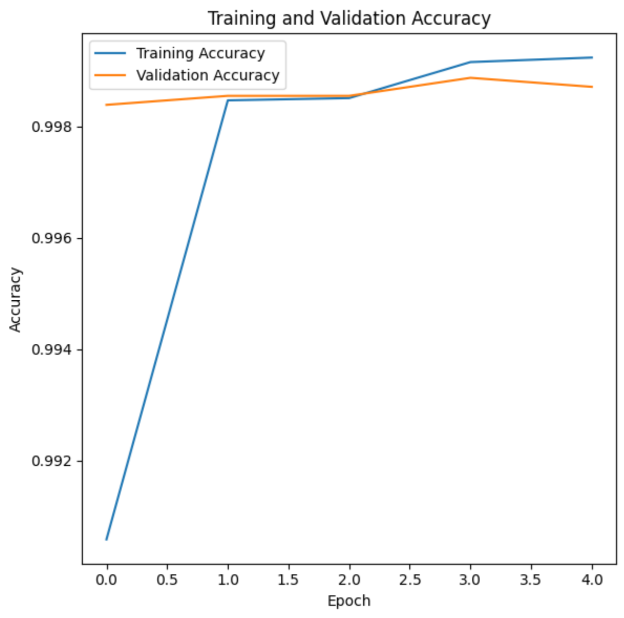


```
Classification Report (Test)  :

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3460
           1       1.00      1.00      1.00      4270

    accuracy                           1.00      7730
   macro avg       1.00      1.00      1.00      7730
weighted avg       1.00      1.00      1.00      7730
```

5. Evaluate Visualization and Summarization

Accuracy and Loss curve



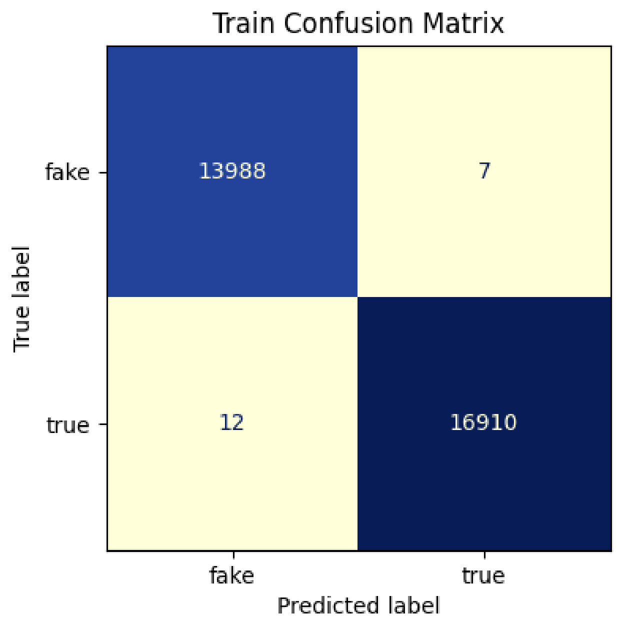


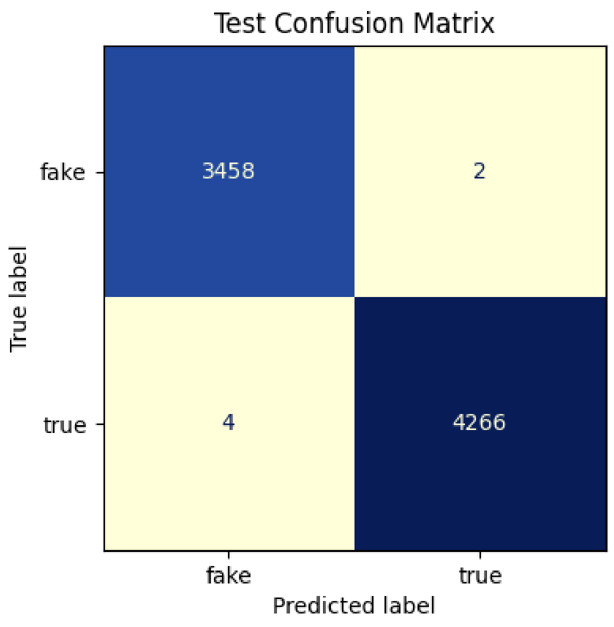
- **Loss Plot:**
  - ➢ The x-axis represents the epoch, which is one iteration over the entire training dataset.
  - ➢ The y-axis represents loss. Loss is a numerical value used to measure how well a model performs on a given task. Lower loss indicates better performance.
  - ➢ The two lines on the plot show the training loss (blue) and validation loss (orange) over a number of epochs.
  - ➢ The training loss starts high and decreases steadily as the model trains. This suggests the model is learning to perform the fraud detection task.

- **Accuracy Plot:**
  - ➢ The x-axis represents the epoch.

- ➢ The y-axis represents accuracy, a measurement of how often the model makes correct predictions. A higher value indicates better performance.
- ➢ The two lines on the plot show the training accuracy (blue) and validation accuracy (orange) over a number of epochs.
- ➢ Both the training accuracy and validation accuracy start high and increase slightly over time. The training accuracy is consistently higher than the validation accuracy throughout.

<u>Confusion Matrix</u>





- **Training Matrix:**
  - ➢ Ideally, a high number in the TP and TN cells, indicating the model is learning to distinguish fraudulent from legitimate news articles on the training data.

- **Testing Matrix:**
  - ➢ A high number in TP and TN cells here is even more crucial, as it generalizes the model's performance to unseen data and reflects its effectiveness in real-world fraud news detection.