Abstract. In recent years, with the development of technology across the globe, people have been able to gain access to information and news more easily than ever before. For instance, by using their mobile phones, people can get daily news from anywhere in the world at any time. As a result, news plays a huge role in people's daily lives and can have an impact on many different aspects of their lives. Nevertheless, there are media outlets that create false news in order to attract an audience. The problem with these howls is that it is difficult to distinguish between real news and fake news.

The purpose of this study was to develop a model to distinguish between news, which included several types of models, including models LSTM and Bi-LSTM as deep learning models, and Logistic regression as a traditional model. As a final step, comparing the accuracy of news articles provides an appropriate criterion for determining whether or not a news article is accurate. After collecting information from a number of websites, the models were constructed and run, and the results were compared. With an accuracy rate of 98%, the model has been selected as the most accurate model[1].

Keywords: Natural Language Processing, Deep Learning, Classification, Fake News Detection, LSTM.

1 Problem Statement

A long history of information and data has led to the creation of many jobs and the employment of many people in society. As a result, the existence of various information and data has a long history in human life. Now, taking into account the speed of news dissemination in the mass media, it is possible to achieve the impact of unreal news on the lives of people. The speed at which news spreads from one person to another also makes fake news extremely easy for the audience to conceal among the correct news. This is because they are unable to recognize it. It is for this reason that one of the concerns these days is spreading the word about their distinction. Solutions have been developed in order to achieve this.

The purpose of spreading fake news in the media is to use attractive and trending topics in society in order to increase their audience in order to increase the number of people who read the fake news, and this has been a technique used with fake news in the media for years. As a result of this process, the performance of society is influenced by the speed at which news is reported.[2]

2 Implementation steps

2.1 Libraries

As part of our study, we have attempted to use a few of the popular Python software libraries, as follows:

- NumPy version: 1.23.0
- Pandas version: 1.5.2
- Matplotlib version: 3.6.2
- Seaborn
- NLTK version: 3.7
- Sklearn
- Tensorflow version: 2.11[3]

2.2 Importing the dataset

The initial data will be called fake and real in the table at this point in time

2.3 Dataset

Among data in this file are two files, one of them contains fake news, another one contains real news. (Fig.1).

Fig. 1.Dataset Details

```
The shape of the  data is (row, column):(23481, 4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23481 entries, 0 to 23480
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   title    23481 non-null  object
 1   text     23481 non-null  object
 2   subject  23481 non-null  object
 3   date     23481 non-null  object
dtypes: object(4)
memory usage: 733.9+ KB
None
```

3 Preprocessing and Cleaning Data

3.1 Creating the target column

In this case, we are going to attempt to create a target column that would take into account both fake news as well as true news. When it comes to fake news, we will denote the target value as '0', while for true news we will denote it as '1.' (Fig. 2).

Fig. 2. Target column

```
#Target variable for fake news
fake_news['output']=0

#Target variable for true news
true_news['output']=1
```

3.2 Concatenating title and text of news

In order to classify news, both the title and the text need to be considered. Treating the title and the text as separate columns does not yield any benefits, so concatenate both columns in both datasets. (Fig. 3)

Fig. 3 Concatenating title and text of news.

```
#Concatenating and dropping for fake news
fake_news['news']=fake_news['title']+fake_news['text']
fake_news=fake_news.drop(['title', 'text'], axis=1)

#Concatenating and dropping for true news
true_news['news']=true_news['title']+true_news['text']
true_news=true_news.drop(['title', 'text'], axis=1)

#Rearranging the columns
fake_news = fake_news[['subject', 'date', 'news','output']]
true_news = true_news[['subject', 'date', 'news','output']]
```

3.3 Appending two datasets

To train the data with the selected models, we must provide all the data as a file. The objective of this study is to assemble both real and fake news data, preprocess them, and conduct exploratory data analysis of them. (Fig. 4)

	subject	date	news	output
0	News	2017-12-31	Donald Trump Sends Out Embarrassing New Year'...	0
1	News	2017-12-31	Drunk Bragging Trump Staffer Started Russian ...	0
2	News	2017-12-30	Sheriff David Clarke Becomes An Internet Joke...	0
3	News	2017-12-29	Trump Is So Obsessed He Even Has Obama's Name...	0
4	News	2017-12-25	Pope Francis Just Called Out Donald Trump Dur...	0
...	...	...	...	...
21412	worldnews	2017-08-22	'Fully committed' NATO backs new U.S. approach...	1
21413	worldnews	2017-08-22	LexisNexis withdrew two products from Chinese ...	1
21414	worldnews	2017-08-22	Minsk cultural hub becomes haven from authorit...	1
21415	worldnews	2017-08-22	Vatican upbeat on possibility of Pope Francis ...	1
21416	worldnews	2017-08-22	Indonesia to buy $1.14 billion worth of Russia...	1

44888 rows × 4 columns

Before EDA, it is necessary to remove stop words and punctuation from the document. This is in order to enhance the accuracy of the display of the data at the next stages.

```
def review_cleaning(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

```
stop = stopwords.words('english')
clean_news['news'] = clean_news['news'].apply(lambda x: ' '.join([word for word in x.spl
it() if word not in (stop)]))
clean_news.head()
```

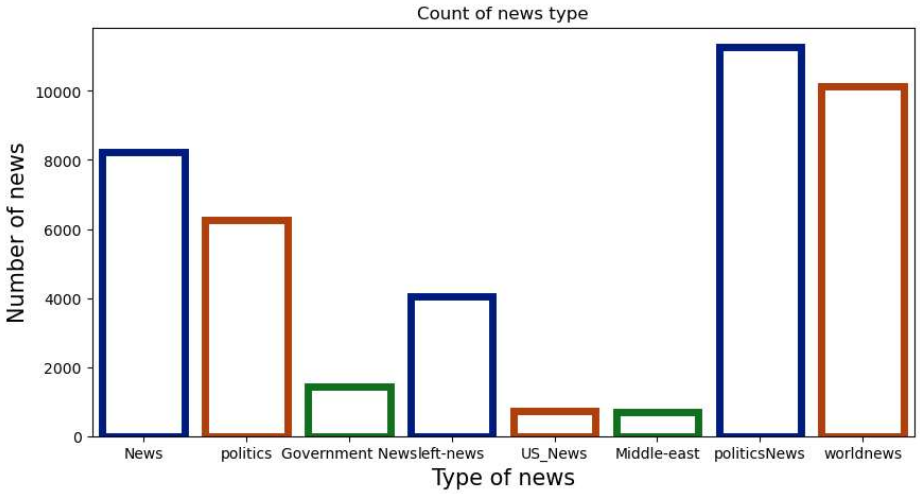
4 Exploratory Data Analysis

4.1 Number of the news subject

There were a large number of articles about political news, and it is evident from the news categories that most of the news is related to the political news. Any category in

the news category can be considered similar to another, such as news, politics, and government news, as they are all deemed the same as one. (Fig. 6.)

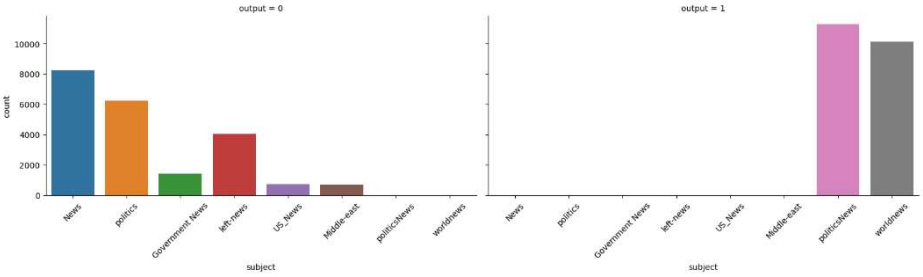
Fig. 5. Type of News



4.2 Number of news subject based on true or fake

In spite of the fact that false news dominates all categories except for politics and world news, the truth tends to be pushed to the politics and world news categories, where the numbers are more, since they dominate all other categories. However, this does not necessarily mean that the model is suitable because the dataset is of low quality. (Fig. 7.)

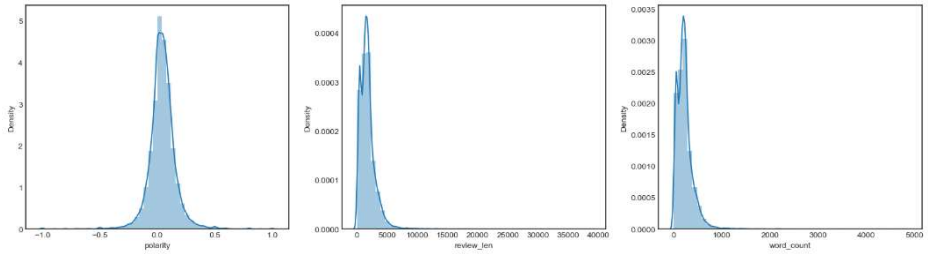
Fig. 7.



4.3 Deriving new features from the news

As part of the process of determining the polarity of a news article, a measure that is used to determine the sentiment of the article is taken into consideration. The length of an article is defined as its words and spaces. This can also be referred to as the word count, which is how many words are in an article. (Fig. 8.)

Fig.8. Deriving new features from the news

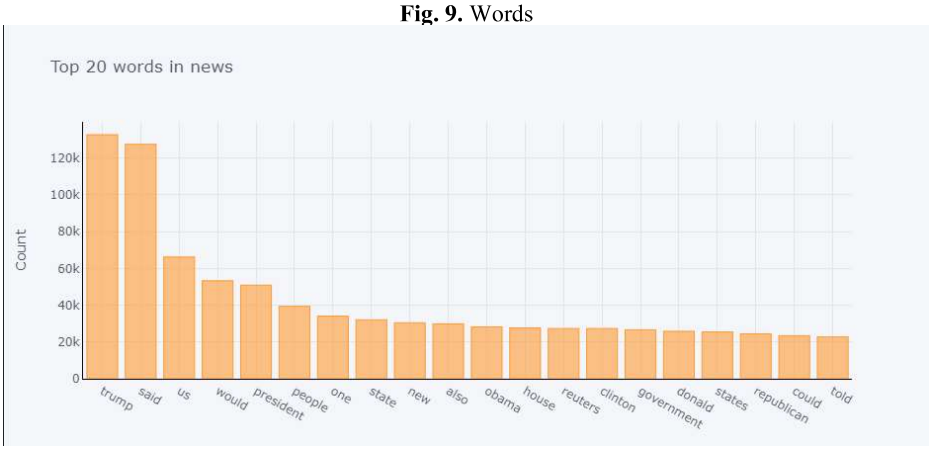


4.4 Grams Analysis

list of the 20 most important words.

The top 20 stories of the week mainly deal with Trump's presidency, followed by Barack Obama as a close second. This is not a surprise because the news is generally provided by Reuters, which is part of the American government. (Fig. 9.)

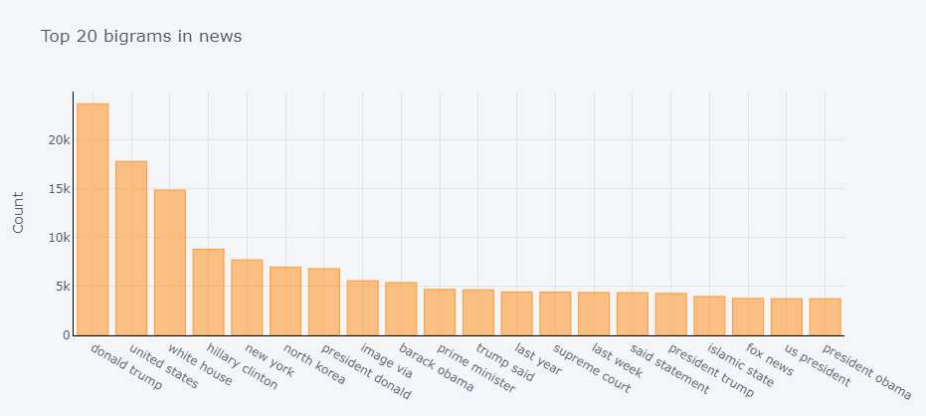
Fig. 9. Words



Two words that are trending in the news at the moment.

There are many news sources online nowadays, but they are usually all related to the time frame of the elections that took place in the United States of America. It follows that there are also more fake news sites during this period, as well as news about North Korea and Fox News. (Fig. 10.)

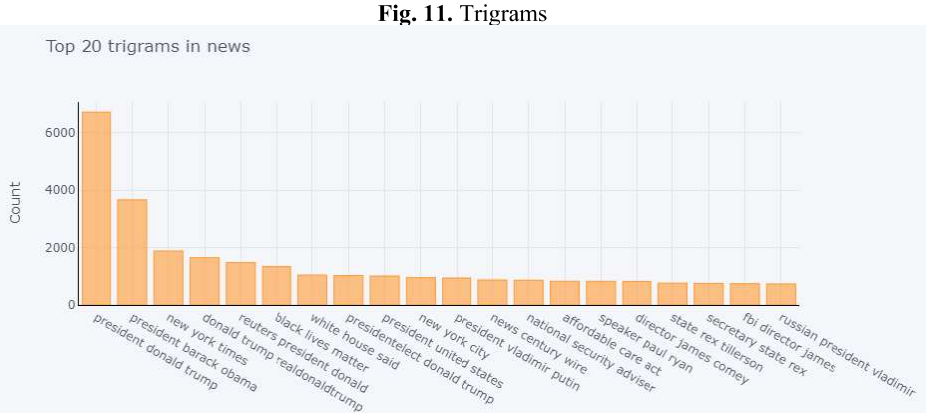
Fig. 10. Bigrams



Top 3 words in the news.

As a result of the death of Floyd, there were a lot of fake news stories that took place that revolved around his death. The rest of the news is about US politics. (Fig. 11.)

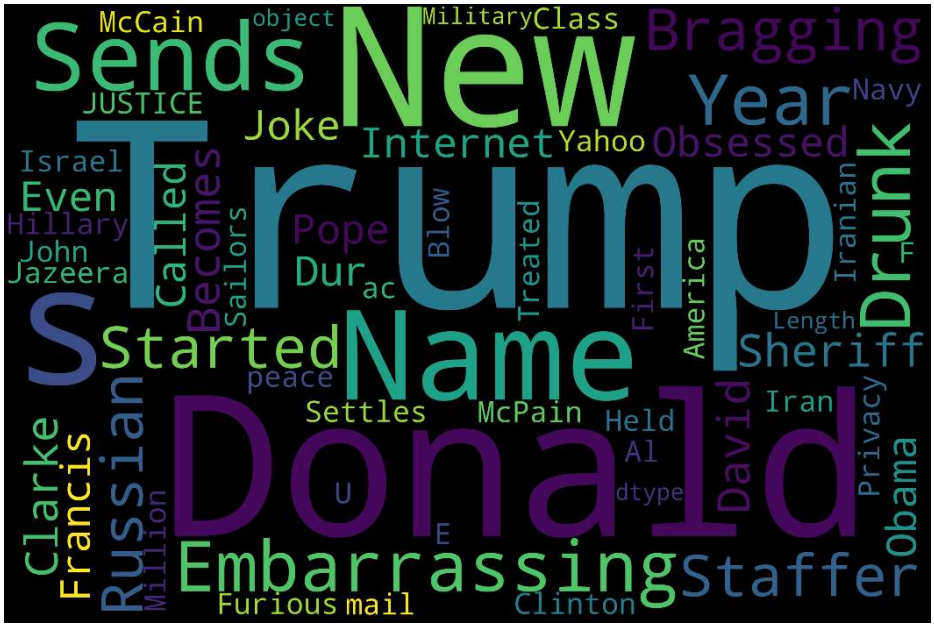
Fig. 11. Trigrams



4.5 Word Cloud of Fake and True News

Fake news.

Fig. 12. Fake news word cloud



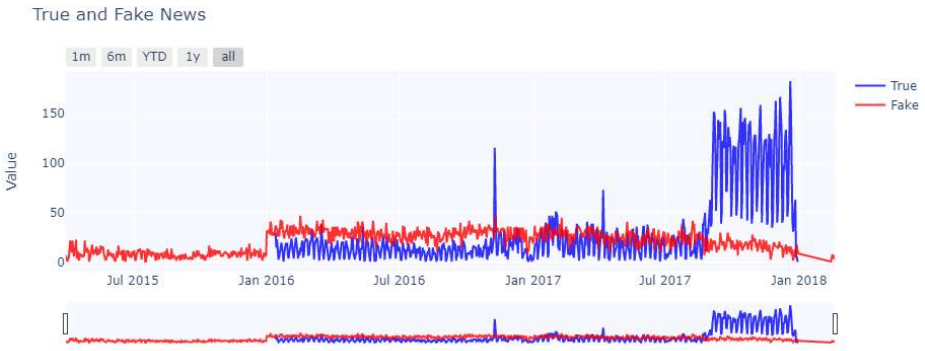
True news.

Fig. 13. Fake news word cloud

4.6 Time series analysis

Real news has grown even more in 2017, as it is seen in much higher numbers. This is a positive sign. Given the fact that we don't have real news data for the entire year 2015, we can see that there are more fake news than real news. This suggests that the classification of fake news is likely to be more accurate than the classification of real news in general.

Fig. 13. 4.6 Time series analysis



5 Stemming and Lemming

When inflected words are taken and derived from their inflected forms, the root words can be derived from them. This can be done by using the inflected form of the word to deduce its root word. Our goal here is to convert the words in the reviews into their root words, which is what we will be doing. It is imperative to note that the root words we extract from the reviews must carry no semantic meaning at all. Therefore, we will convert those words into their root words. It is a process of transforming a word into a root word that can be used with semantic meaning. As a result, I prefer to use stemming rather than using this method since it is a shorter one and less time-consuming. (Fig. 15.)

Fig. 15.

```
stop_words = set(stopwords.words("english"))
#Performing stemming on the review dataframe
ps = PorterStemmer()

#splitting and adding the stemmed words except stopwords
corpus = []
for i in range(0, len(news_features)):
    news = re.sub('[^a-zA-Z]', ' ', news_features['news'][i])
    news= news.lower()
    news = news.split()
    news = [ps.stem(word) for word in news if not word in stop_words]
    news = ' '.join(news)
    corpus.append(news)
```

5.1 TFIDF(Term Frequency — Inverse Document Frequency)¶

There is a technique known as frequency-inverse document frequency or TF-IDF that quantifies the number of words in a document by quantifying word frequency information, which consists of word frequency information. In most cases, weights assigned to words within a document are used by information retrieval and text mining applications. This is in order to determine the importance of those words in that document. There is an extremely wide variety of methods that can be applied to determine the importance of words within a document using this type of method. (Fig. 16.)

Fig. 16. TFIDF

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
# TF-IDF feature matrix
X= tfidf_vectorizer.fit_transform(news_features['news'])
X.shape
```

```
(44888, 5000)
```

5.2 Train-Test split

Our aim is to train the data with terms based on the existing terms that are available in this database, Trian and Test, which are divided into two categories. (Fig. 17.)

Fig. 17. Splitting Data to Test and Trian

```
## Divide the dataset into Train and Test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=
0)
```

6 Model Building

At this stage of the project, we have divided the usable models into two categories, which are traditional models and deep learning methods. This is in order to be able to conduct a meaningful comparison between the two types of methods that currently exist.

6.1 Traditional models

. The following four methods will be discussed in this section. Furthermore, the performance of each method will be compared with each other so that we can identify the most appropriate model on the basis of accuracy.

- **Logistic regression Test** [4]
- **Decision Tree Test:**[5]
- **KNN(K-Nearest Neighbours)Test**[6]
- **Naive Bayes Test:**[7]

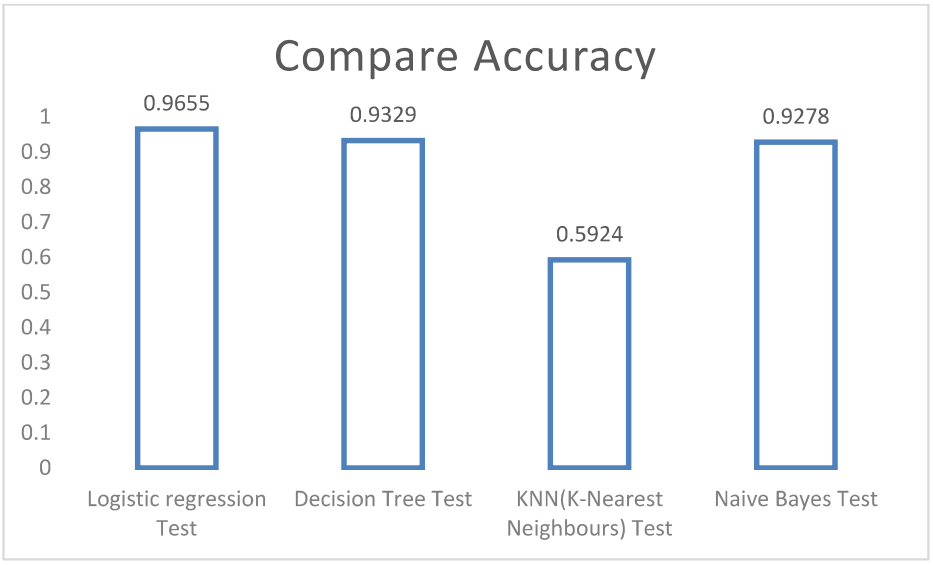
Fig. 18. Traditional models

```
#creating the objects
logreg_cv = LogisticRegression(random_state=0)
dt_cv=DecisionTreeClassifier()
knn_cv=KNeighborsClassifier()
nb_cv=MultinomialNB(alpha=0.1)
cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'Naive Bayes'}
cv_models=[logreg_cv,dt_cv,knn_cv,nb_cv]

#Printing the accuracy
for i,model in enumerate(cv_models):
    print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X, y, cv=10, scoring ='accuracy').mean()))
```

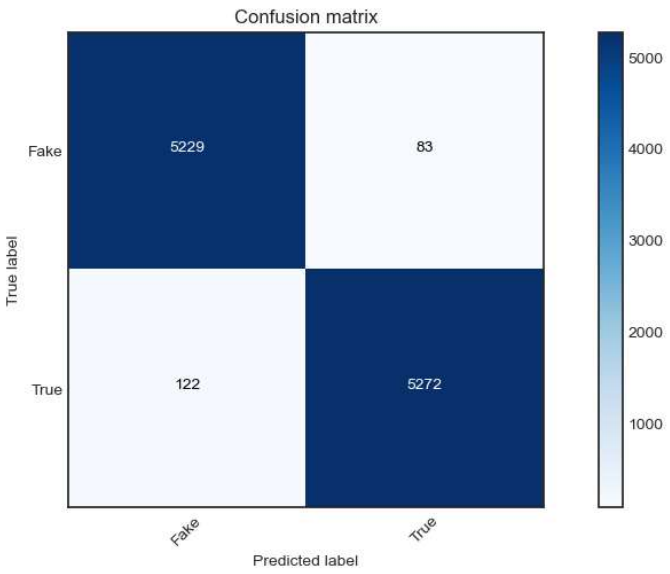
```
Logistic Regression Test Accuracy: 0.9655792542189378
Decision Tree Test Accuracy: 0.9329580858161748
KNN Test Accuracy: 0.5924757164824971
Naive Bayes Test Accuracy: 0.9278185167280396
```

As a result, we can now look at the accuracy of the above models in detecting news. As you can see from the graph below, it can be seen that Logistic regression Test performed the most efficiently among all the models as far as detecting news is concerned with 0.9655 accuracy.



Moreover, if we examine the confusion matrix, we can see that this model has an operator. (Fig. 19).

Fig. 19. Confusion Matrix



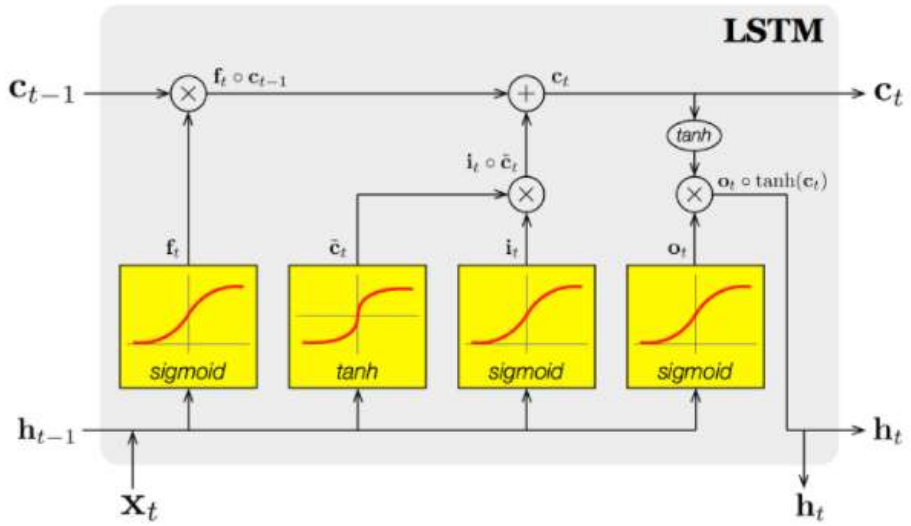
6.2 Deep Learning Models

We are going to use LSTM (Long Short-Term Memory) and Bi-LSTM (Bidirectional long-short term memory) models for training in this section.

LSTM (Long Short-Term Memory)

Using Deep Learning methods, we need three parameters: vocabulary size, sentence length and vector feature. These parameters will all be applied as input to the model through a layer called embedding. Adam's method has also been applied to optimize this model, and an activation function is required for the initial layer. In this model, the sigmoid function is used as the activation function. (Fig. 20).[8]

Fig. 10. Long Short-Term Memory



One hot for Embedding layers.

The first thing we need to do before creating layers is measure the vocabulary size of each layer. This will enable us to be sure to utilize it correctly in the future. The index of each word in the corpus is taken from the vocabulary size of each sentence. This is in order to embed layers within those sentences when the words are encoded within them. It is imperative to note that we will encode the sentences in the corpus in one go. This vocabulary size is therefore necessary.[9]

Padding embedded documents

As a matter of fact, there are many neural networks that are designed to work with inputs of the same size and shape that they require. However, when we use texts as inputs, not all of the sentences are the same length. In addition, padding must be used

to ensure all of the inputs are of the same size. For example, some sentences naturally are longer or shorter than others. Thus, padding is needed in order to ensure that the inputs are all of the same size as they are naturally. The padding here is performed using the pad_sequence() function that uses 5000 as the common length of the sentences. To make the sentences equal length, we will also add zeroes before the sentences in order to achieve even spacing. [10]

Fig. 11. One-Hot and Embedding

```python
#Setting up vocabulary size
voc_size=10000

#One hot encoding
onehot_repr=[one_hot(words,voc_size)for words in corpus]
```

Python

Padding embedded documents

```python
#Setting sentence length
sent_length=5000

#Padding the sentences
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
```

Python

Code implementation in Python.

Fig. 12. LSTM coding

LSTM Model

```python
#Creating the lstm model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Dropout(0.3))
model.add(LSTM(100)) #Adding 100 lstm neurons in the layer
model.add(Dropout(0.3))
model.add(Dense(1,activation='sigmoid'))

#Compiling the model
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

Python

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 5000, 40)          400000

 dropout (Dropout)           (None, 5000, 40)          0

 lstm (LSTM)                 (None, 100)               56400

 dropout_1 (Dropout)         (None, 100)               0

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 456,501
Trainable params: 456,501
Non-trainable params: 0
_________________________________________________________________
None
```

The model was implemented in Python and once the model was implemented, the data was used to train the model. In the model training stage, there are 10 epochs, in which the accuracy of the model is calculated at each epoch and finally, based on the accuracy of each epoch, the accuracy of the entire model is calculated.

Evaluation of model.

As a result of using the accuracy_score command and the information obtained from the previous steps, it was found that the overall accuracy of the model was equal to 0.9816, which shows that the model can correctly distinguish whether the news is real or fake in 98% of cases. (Fig. 24).

Fig. 2413. Evaluation

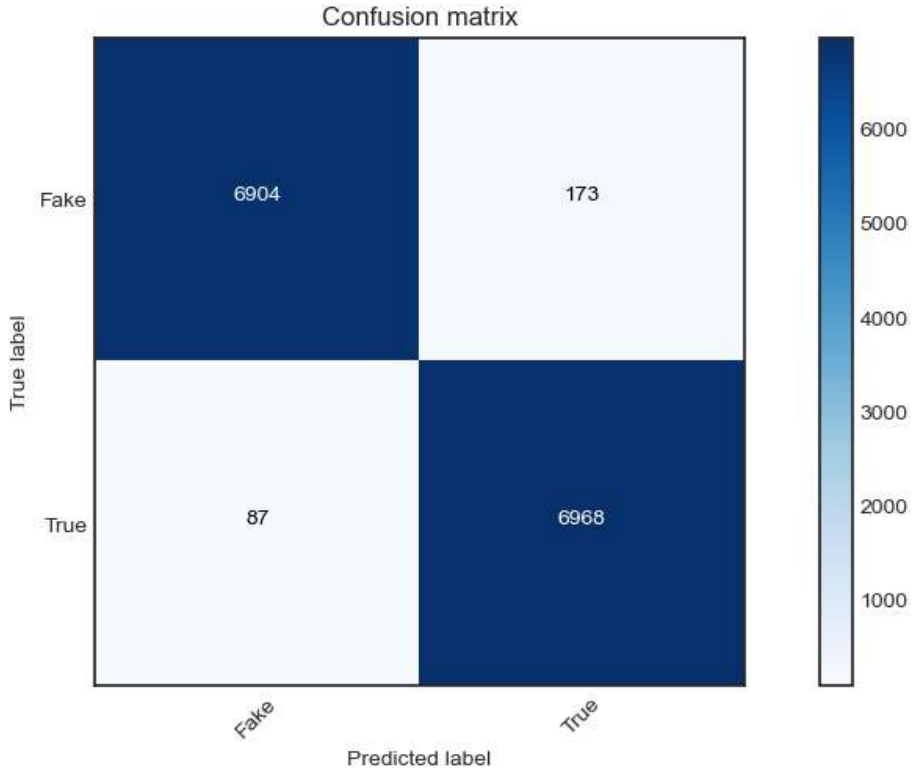
```
#Checking for accuracy
accuracy_score(y_test,y_pred)
```

Python

```
0.9816020379281064
```

It is also significant to note that the confusion matrix confirms that the model is accurate. (Fig. 25).

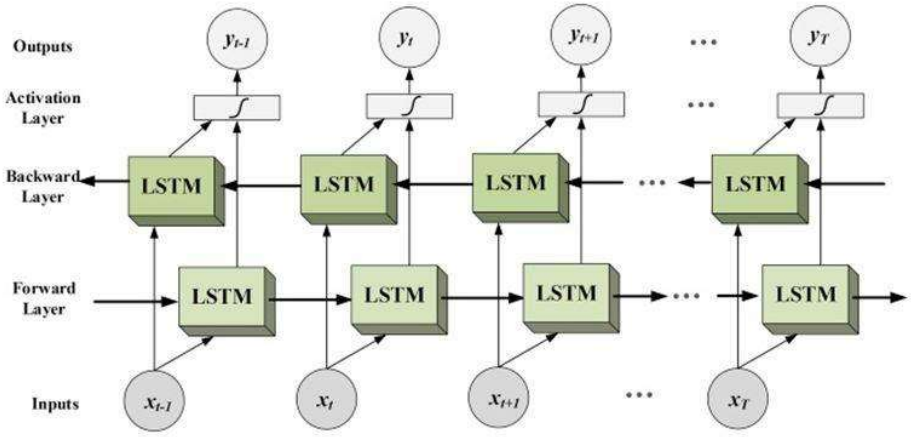
Fig. 25. Confusion Matrix



Bi-LSTM (Bidirectional long-short term memory)

In fact, the Bi-LSTM method is a method in which two LSTM models are used, one in step and the other in reverse. This causes a large increase in the amount of information that is incorporated into the model and as a result, the amount of model error decreases. (Fig. 26).[11]

Fig. 26. Bidirectional Long Short-Term Memory



Code implementation in Python.

Fig. 27. Bi-LSTM Coding

Bidirectional LSTM

```
# Creating bidirectional lstm model
embedding_vector_features=40
model1=Sequential()
model1.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model1.add(Bidirectional(LSTM(100))) # Bidirectional LSTM layer
model1.add(Dropout(0.3))
model1.add(Dense(1,activation='sigmoid'))
model1.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model1.summary())
```

Python

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 5000, 40)          400000

 bidirectional (Bidirectiona  (None, 200)              112800
 l)

 dropout_2 (Dropout)         (None, 200)               0

 dense_1 (Dense)             (None, 1)                 201

=================================================================
Total params: 513,001
Trainable params: 513,001
Non-trainable params: 0
_________________________________________________________________
None
```

The model was implemented in Python and once the model was implemented, the data was used to train the model. In the model training stage, there are 10 epochs, in which the accuracy of the model is calculated at each epoch and finally, based on the accuracy of each epoch, the accuracy of the entire model is calculated.

Evaluation of model.

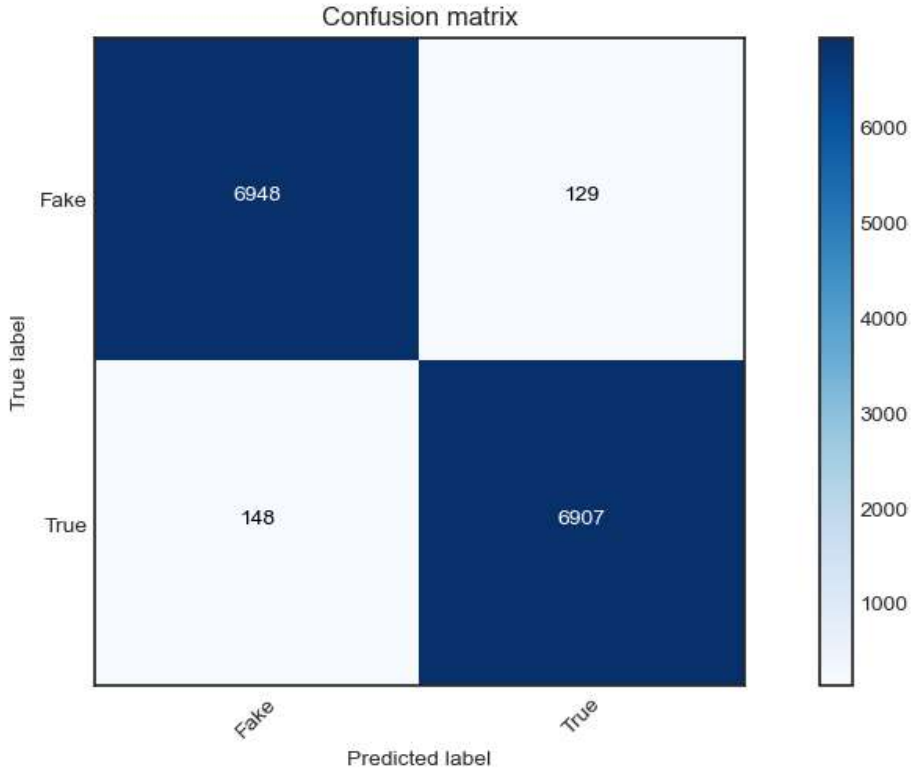
As a result of using the accuracy_score command and the information obtained from the previous steps, it was found that the overall accuracy of the model was equal to 0.9803, which shows that the model can correctly distinguish whether the news is real or fake in 98% of cases. (Fig. 28).

Fig. 28. Evaluation

```
[47]  #Calculating Accuracy score
      accuracy_score(y_test,y_pred1)
                                                                    Python
...   0.9803990942541749
```

It is also significant to note that the confusion matrix confirms that the model is accurate. (Fig. 29).

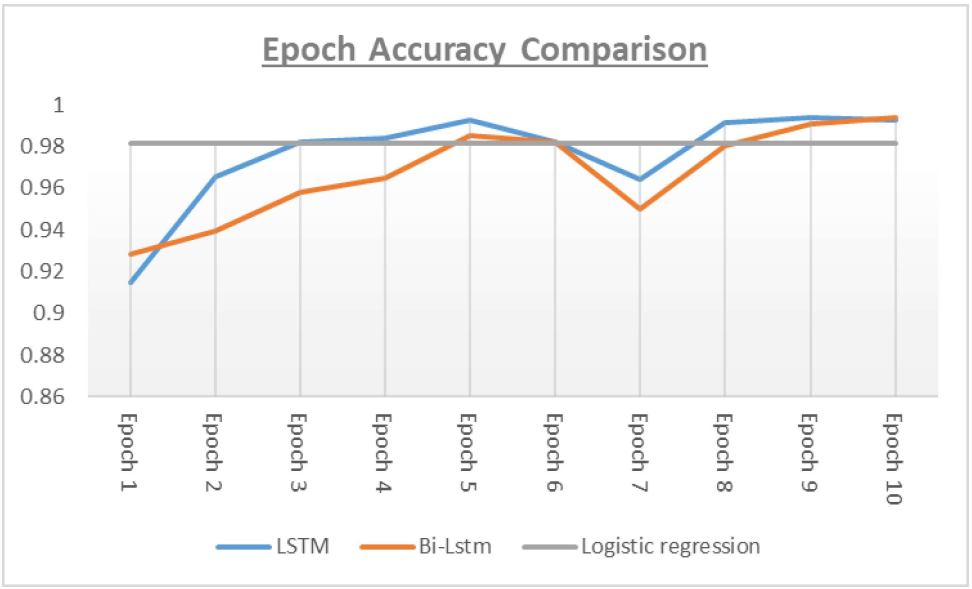
Fig. 29. Confusion Matrix



7 Conclusion

The LSTM and Bi-LSTM models are the best performing models in this study, alongside the Logistic Regression Test model, because all three models reach the same level of accuracy when it comes to detecting fake or real news among the studied models.
In deep learning models, both LSTM and Bi-LSTM methods have the same time, which shows that they both have the same behavior during training. The following diagram illustrates how both models behave in the same way during training, which can be seen in Epoch. (Fig. 30).

Fig. 30. Epoch Comparison


Epoch Accuracy Comparison
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Epoch 9
Epoch 10
LSTM
Bi-Lstm
Logistic regression

References

[1] P. Bahad, P. Saxena, and R. Kamal, "Fake News Detection using Bi-directional LSTM-Recurrent Neural Network," *Procedia Comput. Sci.*, vol. 165, no. 2019, pp. 74–82, 2019, doi: 10.1016/j.procs.2020.01.072.

[2] T. K. K. Ho *et al.*, "Deep Learning-Based Multilevel Classification of Alzheimer's Disease Using Non-invasive Functional Near-Infrared Spectroscopy," *Front. Aging Neurosci.*, vol. 14, no. April, pp. 1–16, 2022, doi: 10.3389/fnagi.2022.810125.

[3] "The_Python_Tutorial_zh-cn." https://docs.python.org/3/tutorial/ (accessed Dec. 22, 2022).

[4] V. S. Rawat, "Logistic regression Logistic regression Logistic regression," *Discov. Stat. Using SPSS*, vol. 404, no. 365, pp. 731–735, 2012, doi: 10.1016/B978-0-12-817084-7.00033-4.

[5] G. Saloni, "Decision Tree - GeeksforGeeks," *Geeksforgeeks.org*, 2019. https://www.geeksforgeeks.org/decision-tree/ (accessed Dec. 22, 2022).

[6] "KNN: K-Nearest Neighbors Essentials - Articles - STHDA." http://www.sthda.com/english/articles/35-statistical-machine-learning-essentials/142-knn-k-nearest-neighbors-essentials/ (accessed Dec. 22, 2022).

[7] S. Prabhakaran, "How Naive Bayes Algorithm Works ? (with example and full code)," *Machinelearningplus.com*, 2018. https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/ (accessed Dec. 22, 2022).

[8] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks," pp. 1–42, 2019, [Online]. Available: http://arxiv.org/abs/1909.09586

[9] "Embedding Layers | Kaggle." https://www.kaggle.com/code/colinmorris/embedding-layers/notebook (accessed Dec. 22, 2022).

[10] "Padding Word2Vec Embeddings with Simple Document Encodings | by Russell Jurney | Towards Data Science." https://towardsdatascience.com/padding-sequences-with-simple-min-max-mean-document-encodings-aa27e1b1c781 (accessed Dec. 22, 2022).

[11] "BiLSTM Explained | Papers With Code," *Paperswithcode*. https://paperswithcode.com/method/bilstm (accessed Dec. 24, 2022).