A Study of Machine Learning and Deep Learning Based Models for Fake News Classification

Sandeep Sohal
Department of Computer Science
Yeates School of Graduate Studies
Ryerson University
Ontario, Canada
kaur.sohal@ryerson.ca

***Abstract*—Spread of unreliable information on the internet has always been an important natural language processing (NLP) problem. It is a complex problem and a lot of studies in this area have suggested deep learning-based methodologies to solve this text classification problem. There are a total of 6 models and experiments presented in this research paper. The first 4 models are based on classical machine learning models including logistic regression, decision tree, naïve bayes, and random forest. Deep learning-based models are also implemented including a simple multilayer perceptron (MLP) model and a state-of-the-art NLP model called bidirectional encoder representations from transformers (BERT) model. The models are evaluated based on accuracy, loss, precision, recall, and f1 scores. Two datasets obtained from Kaggle are additionally used to evaluate the performance of the models [1,2]. The best model performance is achieved with the proposed simple MLP model which achieved a validation accuracy of 99% on the second dataset. This research paper starts by going over the significance of the problem and presents statistical information to understand the motivation behind this research. Several state-of-the-art models are also studied under the literature review section and a technical overview of each proposed methodology and experiment is also discussed. The dataset, preprocessing, results, and final analysis of results are also included in this paper.**

Keywords—natural language processing, NLP, text classification, multilayer perceptron, MLP, Logistic Regression, Decision Tree, Naïve Bayes, Random Forest, Kaggle, Bidirectional Encoder Representations from Transformers, BERT.

I. Introduction

Approximately 500 million tweets are sent out on a daily basis and a million other posts are published on other online platforms [3]. The increasing number of online presence has introduced several other problems that are related to an uncontrollable surge in hate speech and inaccurate spread of information. Facebook utilizes RoBERTa deep learning model to detect posts shared by users that are source of hate speech [4]. These types of deep learning models are responsible for detecting and removing these types of content from the internet. Detection of hate speech is a fairly straightforward task as certain words can be used as indicators and algorithms can be trained to detect and remove these posts. Detecting unreliable sources spreading fallacies about a certain topic is a much more complex task as the wording of sentences may not be a good indicator.

Statistical evidence obtained from CBC news website [5] provide proof that almost 90% of Canadians have been misled by unreliable news sources and posts online. Social media is not the only source of false information, other platforms like magazines and media blog posts are also a major contributor of biased and false spread of information on the web. The most common source of fake news on the internet is Facebook which contributes 68% of falsified information [5]. It is very difficult to monitor the spread of fake news and to identify the source. Once an information is posted on social media platforms it is likely that people will read, and instantaneously share this information to their network connections. This leads to an uncontrollable leak of fake news and it is very common that most people believe in these posts. Once this information is shared and it reaches the attention of other users this leads to issues that may spread propaganda and affect the decision and perception of individuals towards a certain topic. There are several reasons why fake news is created and spread. It includes political or business motives to persuade individuals to believe their information and get their attention to benefit from them. Spread of fake news may become the root of an unforeseeable problem and the motivation behind this research is to find a solution to automatically detect fake news and make classification.

Statistics Canada took a survey to identify the cause of misinformation during the COVID-19 pandemic and identify the source of misleading information spread regarding the vaccine [6]. Misinformation during the early stages of the pandemic lead to some serious life-threatening problems due to lack of research. Social media was one source used by individuals to share their opinions and experiences related to the virus [6]. A lot of people delayed and decided not to get vaccinated due to some research they found online, and as a result this led to an increase in the daily cases reported worldwide. Statistics Canada identified news blog posts and websites as the top contributor of fake news related to COVID-19 during the early stages of the virus [6]. Social media and influencers also were reported as top contributors of misleading information during the pandemic [6].

The purpose of this research is to experiment with 6 different approaches to solve the natural language processing (NLP) problem that is the detection of fake news. A comparison of different machine learning and deep learning methods are presented and the results are discussed. There are a total of 4 classical machine learning approaches implemented which include logistic regression, decision tree, naïve bayes, and random forest. The multilayer perceptron and the bidirectional encoder representations from transformers model also known as BERT are the other two deep learning-based models implemented in this research. The models are evaluated based on their accuracy, precision, recall, loss, and f1 scores. The confusion matrix for each experiment is also generated and studied. A total of 2 different datasets are used to evaluate the best model [1,2]. Based on the experiments performed the best model performance is achieved using the MLP model with an accuracy of 99% on the second dataset which is a complex dataset. Although the BERT model did not meet the expected performance on both datasets, it is still capable of performing much more. It is a strong deep learning model that is specifically developed for NLP related problems and it does take longer to train but it has its own preprocessor and encoder which eliminates the need to first clean the text data.

II. Literature Review

Current studies related to this area of research are Convolutional Neural Network (CNN) and deep learning based. This paper tries to go over some of the novel contributions made by going over some technical papers published in recent years. The aim of this research is to examine different models and algorithms and some of the proposed models in this paper are chosen based on a study of current state of the art in this field of research.

Jose et al. [7] analyzed two different models in their research for the purpose of classifying fake news on online social media (OSM) networks. Their proposed approaches are based on stance detection model with logistic regression, and on bi-directional LSTM model. Their stance detection model produced an accuracy of 90.37%, while their proposed bi-directional LSTM model produced the best performance of 93.46% [7]. The authors started by summarizing their online fake news classification problem into 8 different categories that are relevant to the detection of falsified posts on the internet. These 8 categories include: parody in entertainment industry, manipulation of image and video contents, imposter content, sponsored content, false and unreliable contents, and detection of fabricated content [7]. After identifying different ways a user can create and share unreliable contents on the internet, the authors then formulated a solution to identify these types of information in real time on different social media platforms. Their proposed models are trained to detect only false connections and fabricated contents on the internet [7]. They performed nlp specific preprocessing steps such as removal of stop words, tokenization, and lemmatization. Once the dataset was preprocessed, it was then fed as inputs to the proposed models for training and making predictions. Along with the two proposed models, the authors also trained their dataset using decision tree, random forest, multinomial naïve bayes, and SVM. The best performance was achieved with the bi-directional LSTM model proposed [7].

In another research performed by Garg et al. [8], the attention was placed on detecting COVID-19 related fake news by using different machine learning and deep learning models. The performance was evaluated and the best performance was achieved with deep learning models such as LSTM. The bi-LSTM model and the random forest model achieved the best accuracy of 87% [8]. The also used two datasets to evaluate the performance of their proposed models [8].

In 2018 researchers from China conducted experiments focused on performing feature extraction using TF-IDF, Word2Vec, and Word2Vec weighted by TF-IDF and using SVM as the text classifier to make predictions [9]. It is an interesting research and the researchers found that the from these three proposed models, the approach using Word2Vec weighted by TF-IDF achieved the best results [9]. The highest precision score achieved with this model is 92.21% with recall equal to 92.01% and f1 score is equal to 92.02%. The second-best model performance achieved was with the Word2Vec and SVM based model with precision of 92.05% [9].

Ganesh et al. [10] proposed a unique strategy to not only classify texts as fake or real, but to also identify the source of fake news. Once a source is identified, all posts and contents created by this source will be marked as unreliable. Social network graphs are examined and weights in these graphs are also studied to identify an untrustworthy source [10]. To perform classification the authors used classical machine learning algorithms. The decision tree and random forest were proposed and the dataset used in their research is also the same dataset used in this paper. The second dataset described in this paper [2] was used to train and test the models. To evaluate the model accuracy, precision, recall, and f measure scores were calculated. The best accuracy of 96% was achieved with the decision tree model and 93% accuracy was achieved with the proposed random forest model [10].

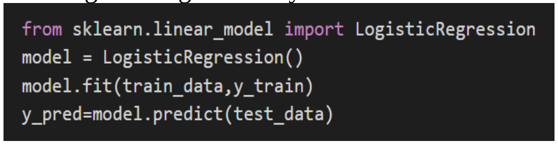
III. Methodology

There are several models and algorithms suggested for solving this NLP problem to predict whether an online content is fake or real. These models and algorithms are typically machine learning or deep learning based. In this research, the performance of both machine learning and deep learning models will be analyzed. A total of 6 models are implemented in this research and 4 of these models are based on machine learning and 2 are deep learning based. This section will go over each model and the technical approach used for implementing these models is described below.

A. *Logistic Regression*

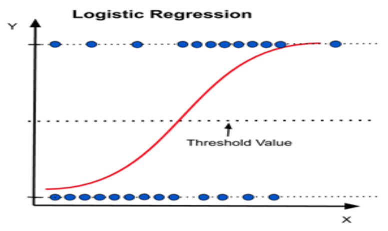
The simplest solution to the binary text classification problem presented in this paper is to use the Logistic Regression model. Once text pre-processing is performed and these texts are vectorized, these machine learning models can be used for making predictions. The implementation of this model is done by using the following function: from sklearn.linear_model import LogisticRegression [11]. The default parameters are utilized and the entire implementation is shown in the figure below.

Figure I: Logistic Regression Python Code



```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(train_data,y_train)
y_pred=model.predict(test_data)
```

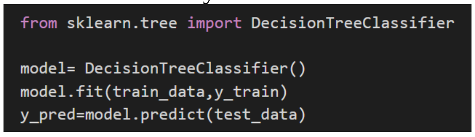
Figure II [12]: Logistic Regression



The logistic regression model tries to fit a sigmoidal curve to separate the two classes and a threshold value is set. This threshold value is typically 0.5 [12]. Anything below this threshold value would belong to the 0 class and results with value greater than 0.5 will belong to the class with label 1 [12]. This relationship is illustrated in the figure above. The logistic regression model implemented in this paper tries to fit the dataset using this sigmoidal curve and use 0.5 threshold value to predict if these texts belong to 0 category which refers to reliable or real news or category 1 which denotes unreliable or fake news.

B. *Decision Tree*

Decision tree is another classical machine learning algorithm that is suitable for performing binary classification tasks. As the name suggests, the algorithm works by forming a tree like structure with the root placed on the top [13]. This tree like structure helps in forming decisions and making predictions. This algorithm can perform both regression and classification tasks [13]. The implementation of this model is also performed using the scikit learn library in python [14]. The following function is used to import the model:
from sklearn.tree import DecisionTreeClassifier [14].
The entire implementation is shown in the code snippet below.

Figure III: Decision Tree Python Code


```
from sklearn.tree import DecisionTreeClassifier

model= DecisionTreeClassifier()
model.fit(train_data,y_train)
y_pred=model.predict(test_data)
```

As shown in the figure the default parameters are used for implementing the model and the results are discussed in later sections. It is a simple algorithm but it might not be the best for complex natural language processing tasks. It definitely will take longer to fit this model on our training data and make new predictions.

C. *Naïve Bayes*

The third machine learning model proposed in this paper is the Naïve Bayes algorithm. This is also a supervised learning algorithm that is based on the Bayes Theorem. It is a probabilistic model used for making classifications and this model is proposed as it works very well with large datasets such as the ones used for this research.

Equation I [15]: Bayes Theorem

$$\underbrace{P(c|x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x|c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

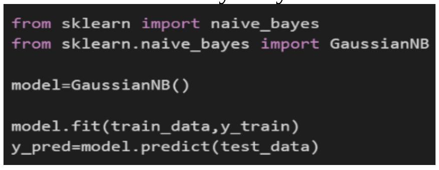
- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

The equation above explains the Naïve Bayes algorithm which tries to calculate the $P(c|x)$, posterior probability [15]. The implementation of this model can simply be done by using the scikit learn library. This library includes different types of Naïve Bayes algorithms including: gaussian, multinomial, complement, Bernoulli, and categorial [16]. This study only concentrates on analyzing the results based on the gaussian naïve bayes model.

Equation II [16]: Gaussian Naïve Bayes

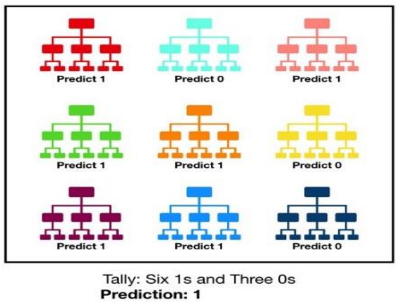
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

In the equation above, the σ_y and μ_y are parameters that are approximated based on the maximum likelihood [16]. The figure shown below illustrates the python code used for implementing this model and the specific libraries that were employed to generate the desired results.

Figure IV: Gaussian Naïve Bayes Python Code


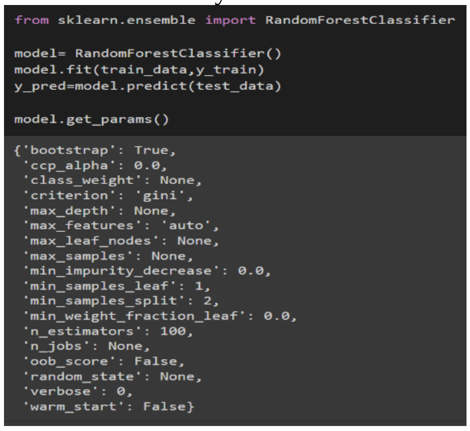
```
from sklearn import naive_bayes
from sklearn.naive_bayes import GaussianNB

model=GaussianNB()

model.fit(train_data,y_train)
y_pred=model.predict(test_data)
```

D. *Random Forest*

Random forest algorithm is built on the idea of decision tree algorithm discussed in the previous section. This algorithm is an ensemble algorithm that works by combining different decision trees to make the final prediction [17]. This makes it much more computationally complex, but it works well with large datasets. It also is intended to perform better than the other machine learning algorithms described above [17]. The algorithm works by first forming several decision trees and the output prediction of each decision tree is computed and a voting procedure is followed to predict the final class [17]. The class with the most votes is chosen as the final predicted label for a particular sample [17]. The figure below describes this algorithm and after performing the voting procedure the final predicted class is selected which is class 1 in this case [17].
Figure V [17]: Random Forest


The implementation code is shown below and the parameters are also shown [18]. The default parameters are kept for performing this experiment.

Figure VI: Random Forest Python Code



```
from sklearn.ensemble import RandomForestClassifier

model= RandomForestClassifier()
model.fit(train_data,y_train)
y_pred=model.predict(test_data)

model.get_params()
```

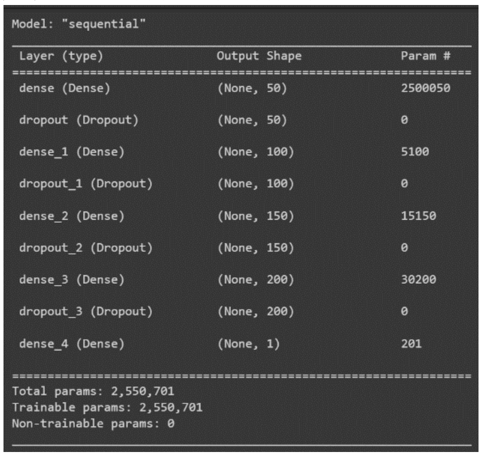
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

E. *Multilayer Perceptron (MLP)*

The multilayer perceptron also known as MLP is an artificial neural network that performs exceptionally well for text classification problems. This method is deep learning based that consists of an input layer, hidden layer, and an output layer. These layers can be designed according to the problem one wants to solve. The proposed MLP model used for solving the fake news classification problem is shown below.

Maximum words of 50000 is given as input to the model and the batch size is equal to 16. The model has 50 neurons in the input layer and the ReLU activation function is employed. The first hidden layer has 100 neurons, followed by 150 neurons in the second hidden layer, and finally in the last hidden layer there are 200 neurons employed. The ReLU activation function is used for each layer except for the output layer which employs sigmoid activation function. The loss function used is binary cross entropy and Adam optimizer with the default learning rate. The model is trained for 30 iterations and validation data is also evaluated. The accuracy, f1 score, recall, loss, and precision are observed at each iteration to monitor the performance of the model. There are a total of 2,550,701 parameters that are trainable.

Figure VII: MLP Model Architecture

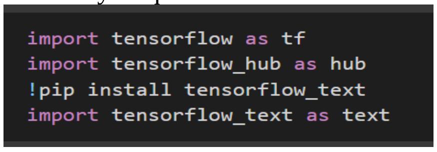


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                2500050

 dropout (Dropout)           (None, 50)                0

 dense_1 (Dense)             (None, 100)               5100

 dropout_1 (Dropout)         (None, 100)               0

 dense_2 (Dense)             (None, 150)               15150

 dropout_2 (Dropout)         (None, 150)               0

 dense_3 (Dense)             (None, 200)               30200

 dropout_3 (Dropout)         (None, 200)               0

 dense_4 (Dense)             (None, 1)                 201

=================================================================
Total params: 2,550,701
Trainable params: 2,550,701
Non-trainable params: 0
_________________________________________________________________
```

F. *Bidirectional Encoder Representations from Transformers (BERT)*

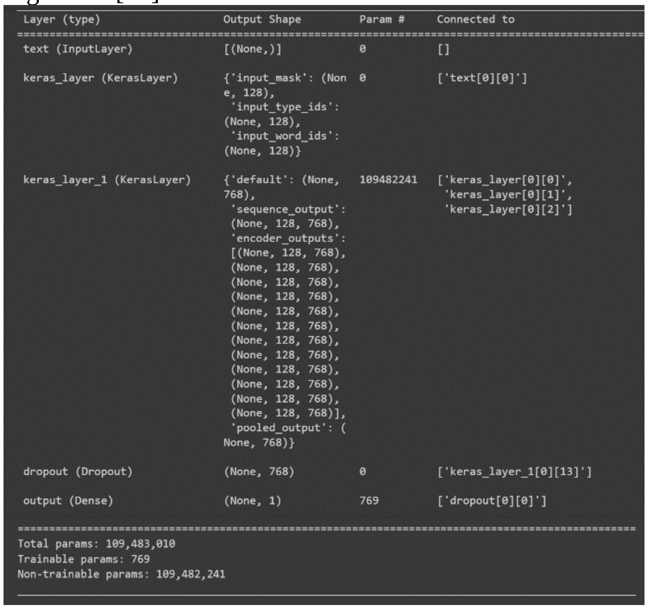
In 2018 Devline et al. [19] from Google AI Language team submitted a paper entitled “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. This is a deep learning-based model that is developed for the purpose of natural language processing and text classification. The model is trained bidirectionally, and this is what separates it from the other models proposed in this research. Models that are bidirectionally trained tend to learn the text data in a deeper way compared to models that are single directionally trained [19]. For this reason, it is considered a better model compared to LSTM’s (Long Short-Term Memory) because these models learn sequentially which does not encapsulate the entire context and some contextual information is lost [19]. LSTM’s have been used for solving NLP problems but due to their training nature the model converges at a much slower pace. Transformers have been used to solve this problem and the BERT model is based on the architecture of transformers [19]. A transformer has an encoder and decoder. The encoder is responsible for reading the input texts and the decoder is responsible for making the predictions. The BERT model only uses the encoder architecture and it stacks the encoders to create the bidirectional encoder representation model that is based on transformers [19]. The BERT model has about 340 million trainable parameters [19]. For this paper, transfer learning was used and implementation of the model is based on the architecture described in the following source [20]. The model was implemented using Tensorflow and Keras library and all the necessary library imports are shown below.

Figure VIII: Library Requirements



```
import tensorflow as tf
import tensorflow_hub as hub
!pip install tensorflow_text
import tensorflow_text as text
```

Figure IX [20]: BERT Model Architecture



```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 text (InputLayer)              [(None,)]            0           []

 keras_layer (KerasLayer)       {'input_mask': (Non  0           ['text[0][0]']
                                e, 128),
                                 'input_type_ids':
                                (None, 128),
                                 'input_word_ids':
                                (None, 128)}

 keras_layer_1 (KerasLayer)     {'default': (None,   109482241   ['keras_layer[0][0]',
                                768),                             'keras_layer[0][1]',
                                 'sequence_output':               'keras_layer[0][2]']
                                 (None, 128, 768),
                                 'encoder_outputs':
                                 [(None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768),
                                 (None, 128, 768)],
                                 'pooled_output': (
                                None, 768)}

 dropout (Dropout)              (None, 768)          0           ['keras_layer_1[0][13]']

 output (Dense)                 (None, 1)            769         ['dropout[0][0]']

==================================================================================================
Total params: 109,483,010
Trainable params: 769
Non-trainable params: 109,482,241
__________________________________________________________________________________________________
```

The BERT model architecture shown above was used for training the text data which was pre-processed using the BERT pre-processor explained in the dataset and pre-processing section below. The BERT encoder can be loaded using the following function: bert_encoder = hub.KerasLayer('https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4') [20].

After going through the pre-processor and encoder, a dropout layer is applied and the dense output layer with 1 neuron for binary classification. The sigmoid activation function is used in the output layer. The model has a total of 109,483,010 parameters, out of which only 769 parameters are trainable. The model is trained for 30 iterations with batch size of 32 and Adam optimizer is used. The loss function used is binary cross entropy and this is observed at each iteration including other performance measures.

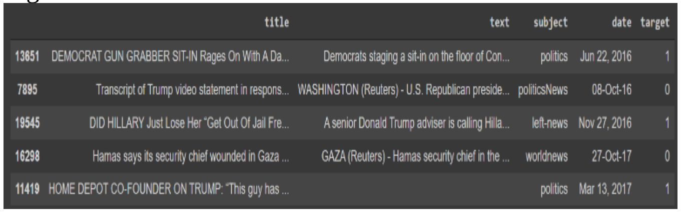
IV. Dataset and Pre-Processing

The proposed models described in the section above are implemented on two different datasets which are described in this section. The structure of the dataset and the pre-processing techniques performed are explained as well.

A. *Dataset 1*

The first dataset is taken from Kaggle and it is called the "Fake and real news dataset" [1]. The dataset contains two comma-separated value files for true and fake news. It occupies 43 MB of storage. A total of 23,503 samples belonging to the fake news category are provided and 21418 samples belonging to the true news category. The features include title, text, subject, and date. A target column was added and all the samples belonging to fake category were labelled as 1 and all the samples belonging to true or real category were labelled as 0. These two .csv files were concatenated and shuffled.

Figure X: Fake and Real News Dataset



	title	text	subject	date	target
13651	DEMOCRAT GUN GRABBER SIT-IN Rages On With A Da...	Democrats staging a sit-in on the floor of Con...	politics	Jun 22, 2016	1
7895	Transcript of Trump video statement in respons...	WASHINGTON (Reuters) - U.S. Republican preside...	politicsNews	08-Oct-16	0
19545	DID HILLARY Just Lose Her "Get Out Of Jail Fre...	A senior Donald Trump adviser is calling Hilla...	left-news	Nov 27, 2016	1
16298	Hamas says its security chief wounded in Gaza ...	GAZA (Reuters) - Hamas security chief in the ...	worldnews	27-Oct-17	0
11419	HOME DEPOT CO-FOUNDER ON TRUMP: "This guy has ...		politics	Mar 13, 2017	1

Exploratory dataset analysis was performed to understand the distribution of the relevant features.

Figure XI: Subject Feature Distribution

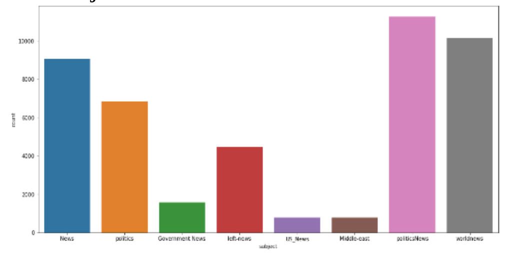


Figure XII: Target Distribution

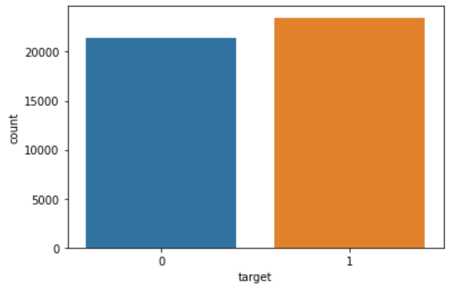
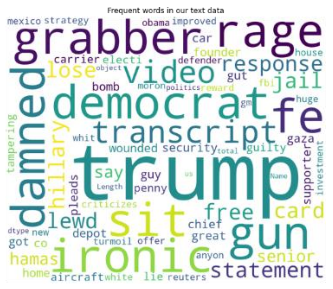


Figure XIII: Word Cloud

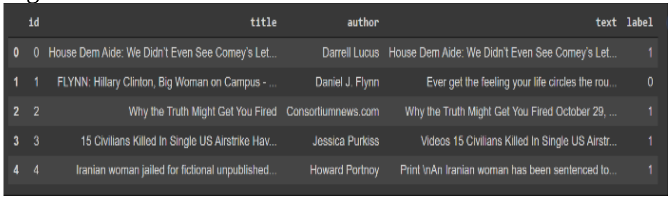


The distribution of the subject feature is highly imbalanced and using imbalanced datasets sometimes leads to problems like overfitting and underfitting. To avoid this problem the subject feature is dropped along with the date feature which does not provide any relevant information that could affect the fake news classification problem.

B. *Dataset 2*

A second dataset is used to verify the results of the models and to evaluate the performance of the models better. This dataset is also obtained from Kaggle and it is entitled "Fake News" [2]. It includes three .csv files and occupies a total storage of 123.81 MB. The train.csv file contains 25,117 samples belonging to reliable and unreliable categories. The test.csv file contains 5,881 samples and the corresponding labels are stored in the submit.csv file. The test and label files were merged and concatenated with the train file.

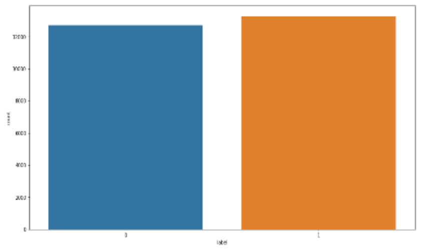
Figure XIV: Fake News Dataset



	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

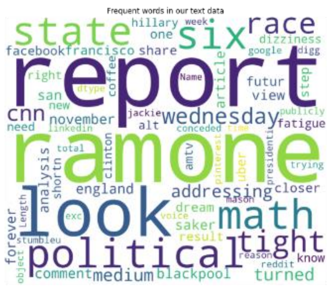
For this dataset only the distribution of label variable can be analyzed and it is illustrated in the figure below.

Figure XV: Label Distribution



Similar to the first dataset, only the text and title features are used for training the models. The other features are removed as they are not significant.

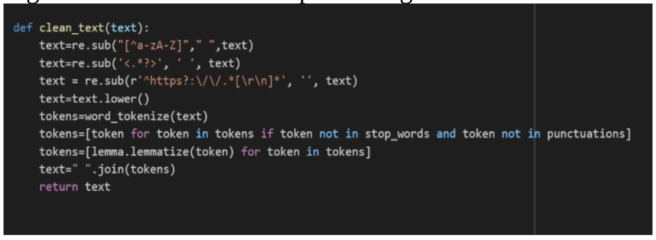
Figure XVI: Word Cloud



C. *Preprocessing*

Data preprocessing is an essential step in performing NLP related problems. This step requires the most time as text data that is scrapped from the web contains a lot of noise. It is important to clean the dataset. The preprocessing steps taken for both datasets are similar. The first step taken was to remove any null values followed by data shuffling and keeping only relevant features from the datasets. The next steps are NLP specific as shown below.

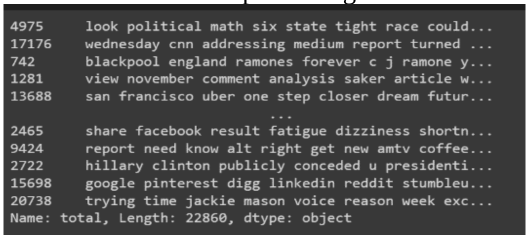
Figure XVII: NLP Text Preprocessing



```
def clean_text(text):
    text=re.sub("[^a-zA-Z]"," ",text)
    text=re.sub('<.*?>', ' ', text)
    text = re.sub(r'^https?:\/\/.*[\r\n]*', '', text)
    text=text.lower()
    tokens=word_tokenize(text)
    tokens=[token for token in tokens if token not in stop_words and token not in punctuations]
    tokens=[lemma.lemmatize(token) for token in tokens]
    text=" ".join(tokens)
    return text
```

The first line of code tells the models to concentrate on alphabets and numbers only. Next, all the html characters are replaced with space. All punctuations are also discarded from the two datasets. These texts are then converted to lower case characters. Tokenization, stop word removal, and lemmatization is also performed to reduce the dimension of the dataset. This will reduce the computational complexity when training these datasets on the models proposed. Since lemmatization performs better than stemming this method is used for getting the root words from the text provided.

Figure XVIII: Text after Preprocessing



```
4975     look political math six state tight race could...
17176    wednesday cnn addressing medium report turned ...
742      blackpool england ramones forever c j ramone y...
1281     view november comment analysis saker article w...
13688    san francisco uber one step closer dream futur...
                              ...
2465     share facebook result fatigue dizziness shortn...
9424     report need know alt right get new amtv coffee...
2722     hillary clinton publicly conceded u presidenti...
15698    google pinterest digg linkedin reddit stumbleu...
20738    trying time jackie mason voice reason week exc...
Name: total, Length: 22860, dtype: object
```

The text after performing all the necessary preprocessing steps is illustrated above. It is now clean and the last step left is to vectorize this text data so it can be used as input to all the machine learning and deep learning models proposed. This step is performed using the scikit learn library by implementing the following function:

from sklearn.feature_extraction.text import TfidfVectorizer.

The dataset is also split and divided into two sets, the training and test set. From the entire dataset 80% of the samples are used for training and the remaining 20% is used for testing the models. The BERT model was preprocessed differently as it has its own preprocessor which can be implemented by using the following function: bert_preprocess = hub.KerasLayer('https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3').

V. Machine Requirements

All the experiments and models described in this paper are performed using google colab and the colab pro+ GPUs are utilized. Utilizing the additional compute resources provided by the colab pro+ subscription helped boost the training speed. High-RAM is also used as it avoids any problems with the RAM that might occur during training. Google colab pro+ provides Tesla P100-PCIE GPU and the device used for training the models is a surface laptop 3 with Intel Core i7 and 16GB RAM.

VI. Experiments and Results

The experiments performed in this research study are all discussed in this section and the results are also illustrated. All models were evaluated based on loss, accuracy, precision, recall, and f1 score. The performance of each model is also evaluated based on two datasets. The model that outperformed all other models on both datasets is selected as optimal and run time analysis of each model is also considered.

A. *Machine Learning Models*

Table I: ML Model Results for 1st Dataset

Machine Learning Models: Results for Dataset 1				
Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Regression	0.9838	0.9889	0.9794	0.9841
Decision Tree	0.9971	0.9966	0.9977	0.9971
Naïve Bayes	0.9644	0.9585	0.9727	0.9656
Random Forest	0.9928	0.9976	0.9883	0.9929

Table II: ML Model Results for 1st Dataset

Machine Learning Models: Results for Dataset 2				
Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Regression	0.8777	0.8757	0.8507	0.8631
Decision Tree	0.8327	0.8101	0.8214	0.8157
Naïve Bayes	0.8182	0.8249	0.7599	0.7911
Random Forest	0.8731	0.8963	0.8141	0.8532

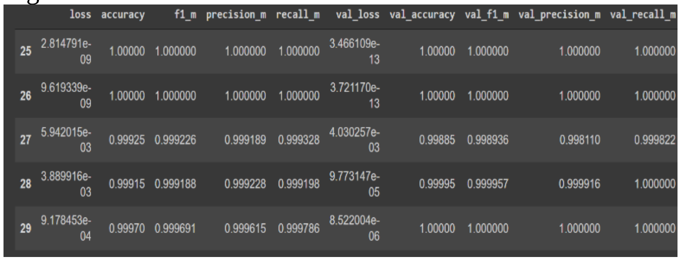
The results obtained after training and testing the 4 machine learning models proposed are shown in the two tables above. The first table shows the results produced using the first dataset described in this paper and the second table includes the results produced using the second dataset. These results show that all the machine learning models performed extremely well with the first dataset, but the performance was negatively affected when the second dataset was used. This could be because the publisher of the first dataset provided a dataset that was preprocessed beforehand. The first dataset being clean and simple, made the job of the machine learning models less complex. The second dataset contains much more noise which is what the models should learn. Text datasets are supposed to be noisy and good machine learning models should be able to learn and fit to these data points. The best performance using the 2nd dataset is achieved with the logistic regression model, but the random forest model provides better precision. Since the task is to predict whether an input text is reliable or unreliable, it is important that all these scores go beyond a certain threshold. The logistic regression model achieved the best result overall from all the proposed machine learning models. The confusion matrix for each experiment is shown in the appendix section below.

B. *Multilayer Perceptron (MLP)*

The MLP model was trained for 30 epochs and it was also evaluated based on the scores achieved with the 1st and the 2nd dataset. The validation accuracy, precision recall, and f1 scores were used to evaluate the model's performance. The binary cross entropy loss function was also observed at each iteration.

All the results are shown in the figures below. It can again be observed that the model performed better with the 1st dataset compared to the 2nd dataset. As a matter of fact, the model achieved a test accuracy of 100% and no signs of overfitting are observed because the training accuracy is also 100% at the 25th iteration shown below. This is a common behavior of neural networks when the given training dataset is very similar to the testing dataset.

Figure XIX: MLP Results of the 1st Dataset



	loss	accuracy	f1_m	precision_m	recall_m	val_loss	val_accuracy	val_f1_m	val_precision_m	val_recall_m
25	2.814791e-09	1.00000	1.000000	1.000000	1.000000	3.466109e-13	1.00000	1.000000	1.000000	1.000000
26	9.619339e-09	1.00000	1.000000	1.000000	1.000000	3.721170e-13	1.00000	1.000000	1.000000	1.000000
27	5.942015e-03	0.99925	0.999226	0.999189	0.999328	4.030257e-03	0.99885	0.998936	0.998110	0.999822
28	3.889916e-03	0.99915	0.999188	0.999228	0.999198	9.773147e-05	0.99995	0.999957	0.999916	1.000000
29	9.178453e-04	0.99970	0.999691	0.999615	0.999786	8.522004e-06	1.00000	1.000000	1.000000	1.000000

Figure XX: Accuracy Plot of the 1st Dataset

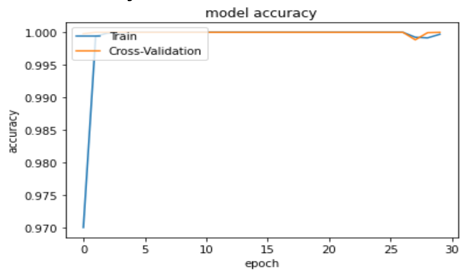


Figure XXI: Loss Plot of the 1st Dataset

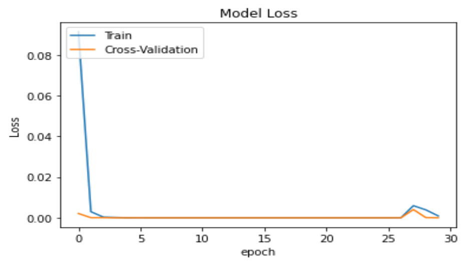
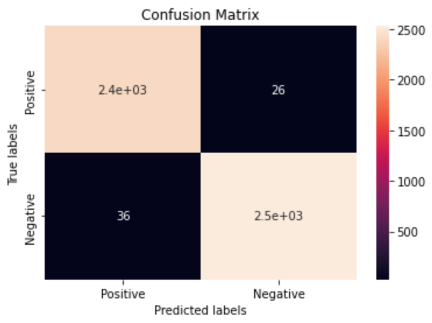
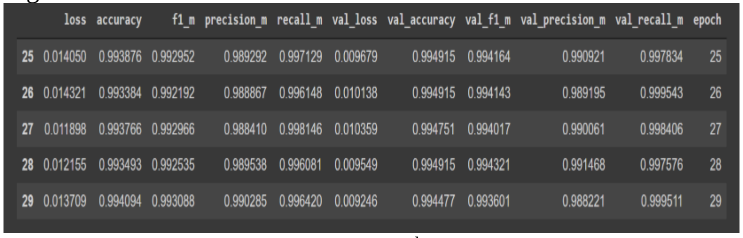


Figure XXII: Confusion Matrix of the 1st Dataset



The accuracy and loss plots shown above indicate that the model converged very fast. It took less than 5 epochs for the model to learn the dataset and train on it. The confusion matrix also shows that only 36 samples were classified as false negatives and 26 samples as false positives.

Figure XXIII: MLP Results of the 2nd Dataset



	loss	accuracy	f1_m	precision_m	recall_m	val_loss	val_accuracy	val_f1_m	val_precision_m	val_recall_m	epoch
25	0.014050	0.993876	0.992952	0.989292	0.997129	0.009679	0.994915	0.994164	0.990921	0.997834	25
26	0.014321	0.993384	0.992192	0.988867	0.996148	0.010138	0.994915	0.994143	0.989195	0.999543	26
27	0.011898	0.993766	0.992966	0.988410	0.998146	0.010359	0.994751	0.994017	0.990061	0.998406	27
28	0.012155	0.993493	0.992535	0.989538	0.996081	0.009549	0.994915	0.994321	0.991468	0.997576	28
29	0.013709	0.994094	0.993088	0.990285	0.996420	0.009246	0.994477	0.993601	0.988221	0.999511	29

Figure XXIV: Accuracy Plot of the 2nd Dataset

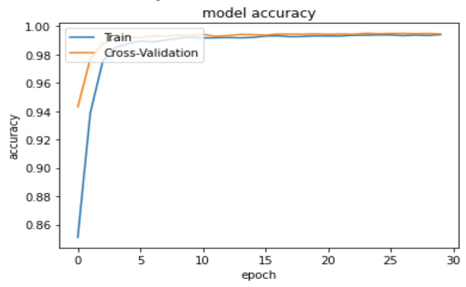


Figure XXV: Loss Plot of the 2nd Dataset

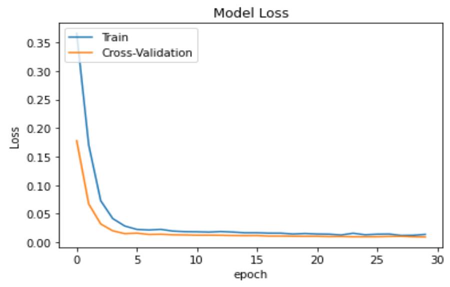
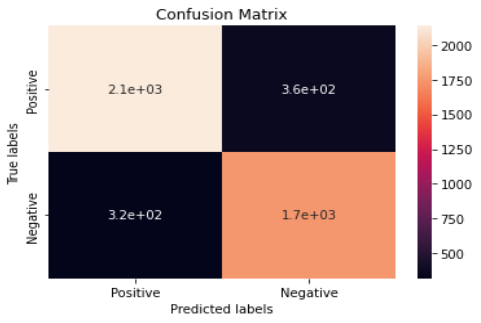


Figure XXVI: Confusion Matrix of the 2nd Dataset

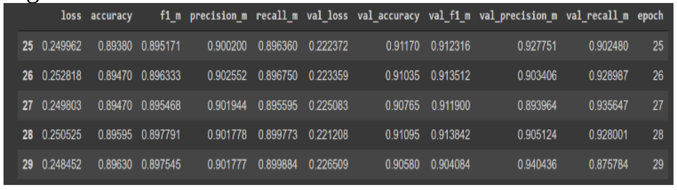


The result after training the MLP model on the second dataset is also very impressive, although slightly less that what was achieved with the first dataset. After running the model for 25 iterations the model converged and reached a test accuracy of 99.49% with a much more complicated dataset. The validation precision, recall, and f1 scores were also in the range of 99% and above. The accuracy and loss plots for the second dataset also show that the model converged very quickly and minimized the loss at the fifth iteration.

C. *Bidirectional Encoder Representations from Transformers (BERT)*

The final experiment performed in this research is the BERT model which is a much more complex deep learning model that was developed for the purpose of performing several natural language processing tasks. It has its own text preprocessor which accepts input texts in the raw form and it also has a pre-defined encoder model which learns the features. After training this model on the first dataset the best test accuracy achieved is 91.170% which is definitely less than what was achieved with the MLP model. This model also took longer to train. The loss and the accuracy plots show that the model did converge after the fifth iteration but since this is a much more complex model it might require more epochs to reach the performance that the MLP model achieved.

Figure XXVII: BERT Results of the 1st Dataset



	loss	accuracy	f1_m	precision_m	recall_m	val_loss	val_accuracy	val_f1_m	val_precision_m	val_recall_m	epoch
25	0.249962	0.89380	0.895171	0.900200	0.896360	0.222372	0.91170	0.912316	0.927751	0.902480	25
26	0.252818	0.89470	0.896333	0.902552	0.896750	0.223359	0.91035	0.913512	0.903406	0.928987	26
27	0.249803	0.89470	0.895468	0.901944	0.895595	0.225083	0.90765	0.911900	0.893964	0.935647	27
28	0.250525	0.89595	0.897791	0.901778	0.899773	0.221208	0.91095	0.913842	0.905124	0.928001	28
29	0.248452	0.89630	0.897545	0.901777	0.899884	0.226509	0.90580	0.904084	0.940436	0.875784	29

Figure XXVIII: Accuracy Plot of the 1st Dataset

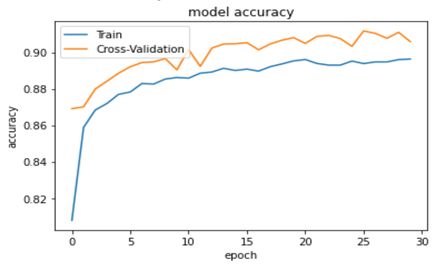


Figure XXIX: Loss Plot of the 1st Dataset

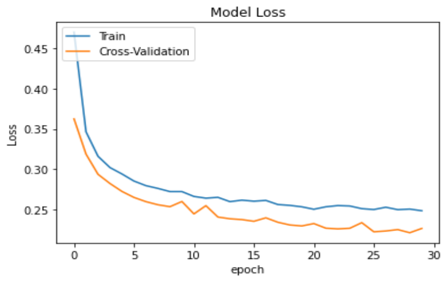
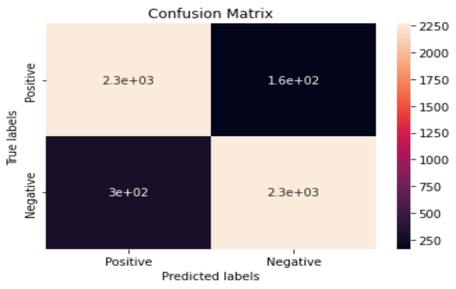
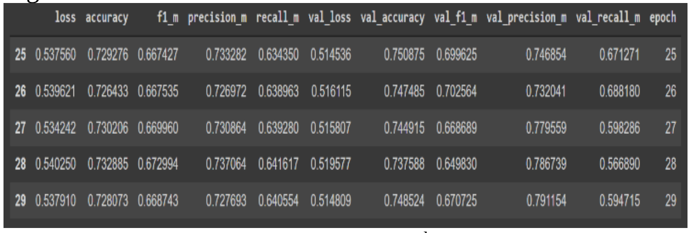


Figure XXX: Confusion Matrix of the 1st Dataset



The results obtained after running this model on the second dataset are also shown below. This model produced results that are even worse than the results obtained with the classical machine learning algorithms proposed above. This was again trained for 30 epochs and the best performance is achieved after the 25th iteration. The model achieved a validation accuracy score of 75.09% and the recall is 67.13%. This might not be the best model for this dataset. Although, it is possible that the model will start producing better results after training it for another 20 iterations making it a total of 50 iterations.

Figure XXXI: BERT Results of the 2nd Dataset



	loss	accuracy	f1_m	precision_m	recall_m	val_loss	val_accuracy	val_f1_m	val_precision_m	val_recall_m	epoch
25	0.537560	0.729276	0.667427	0.733282	0.634350	0.514536	0.750875	0.699625	0.746854	0.671271	25
26	0.539621	0.726433	0.667535	0.726972	0.638963	0.516115	0.747485	0.702564	0.732041	0.688180	26
27	0.534242	0.730206	0.669960	0.730864	0.639280	0.515807	0.744915	0.668689	0.779559	0.598286	27
28	0.540250	0.732885	0.672994	0.737064	0.641617	0.519577	0.737588	0.649830	0.786739	0.566890	28
29	0.537910	0.728073	0.668743	0.727693	0.640554	0.514809	0.748524	0.670725	0.791154	0.594715	29

Figure XXXII: Accuracy Plot of the 2nd Dataset

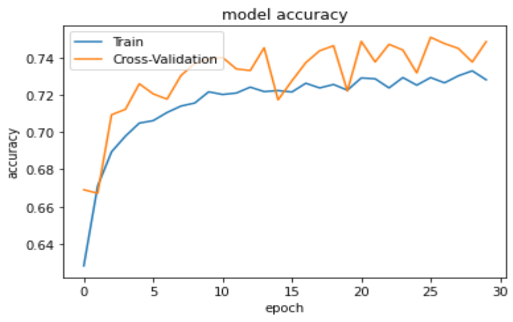


Figure XXXIII: Loss Plot of the 2nd Dataset

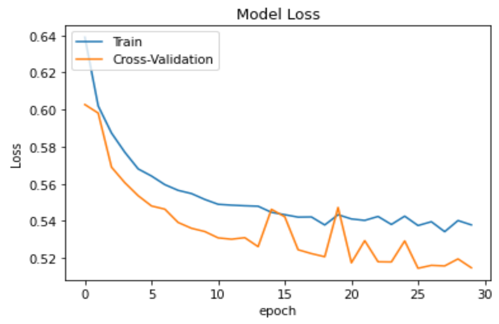
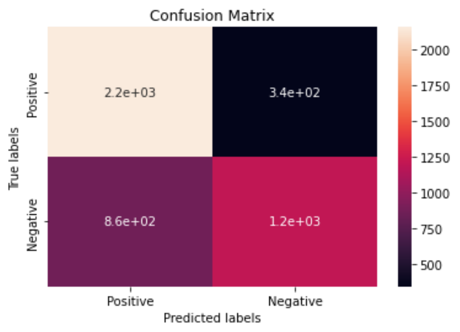


Figure XXXIV: Confusion Matrix of the 2nd Dataset



VII. FUTURE WORK

The models presented in this paper can be extended for real time applications in the future. This study can be used to make classification of texts in real time to avoid the spread of fake news and identify the source immediately. Social network graphs can also be studied to analyze the source of fake news and identify the community affected by the spread of a particular unreliable information on the internet. This is a method similar to what Ganesh et al. [10] proposed in their research which is described in the literature review section. The BERT model can be trained for more iterations to observe any changes in its performance. Another experiment that could be performed in the future is to use the Word2Vec weighted by TF-IDF neural network model for feature extraction and use a classical machine learning algorithm to perform classification. This is a similar approach taken by Zhang et al. [9]. In this paper the TF-IDF vectorizer from the scikit library is used for vectorizing the clean text dataset.

New dataset can be generated by scrapping text data from different online platforms and the MLP and BERT models can be fine-tuned accordingly to obtain more interesting results from this research in the future.

VIII. Conclusion

Fake news classification is an important NLP problem and it is an active area of research in the machine learning and data science community. Working with text data is not a simple task due to its noisy nature. It requires thorough cleaning and preprocessing steps. This makes the classification of texts a computationally complex problem. Machine learning and deep learning-based models are proposed in this research. The problem with machine learning models is that they require a lot more time to train, but these models are capable of learning from a smaller dataset. Deep learning-based models on the other hand require a lot more training data, but these models are capable of generalizing the problem.

In recent NLP papers, deep learning models are suggested. In particular the BERT model introduced by Google has gained a lot of popularity in the field. However, this model does take a lot longer to converge but if the preprocessing step is taken into consideration, then there will not be much difference in the training time. The BERT model has its own built in text preprocessor which accepts raw text data and an encoder. This model is capable of generalizing and learning all types of text data. This complex nature of the model is expected to produce the best results, but based on the experiments performed in this research the simple MLP model outperformed the state-of-the-art deep learning model. It achieved an accuracy of 99% on the second dataset and the BERT model was only able to produce an accuracy of 75.09%. For this reason, the experimental results obtained from this research suggest that the MLP model is optimal for fake news classification task. Training the BERT model for 50 iterations may also produce better results and this bidirectional encoder model is suggested if this research is to be extended for performing real-time fake news detection. This model is capable of generalizing to different forms of text data which makes BERT a better model for all types of datasets.

Overall, it is an interesting area of research and due to the noisy nature of text data, there is always room for improvement. Fine-tuning these models and training on a scrapped dataset from an online source can be done to further improve this study in the future.

IX. Appendix

This section includes the generated confusion matrix for all the machine learning models discussed in the sixth section which is the experiments and results section.

Figure XXXV: Logistic Regression Confusion Matrix for the 1st Dataset

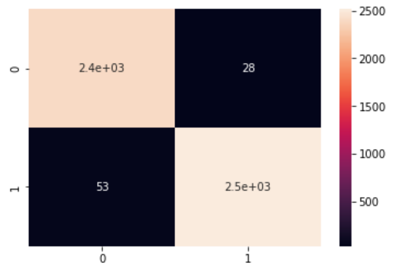


Figure XXXVI: Decision Tree Confusion Matrix for the 1st Dataset

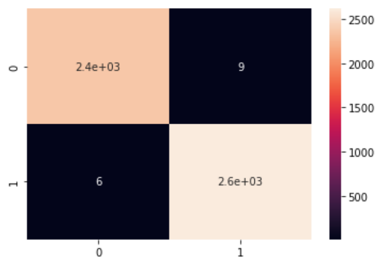


Figure XXXVII: Naïve Bayes Confusion Matrix for the 1st Dataset

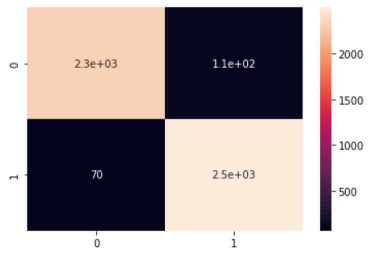


Figure XXXVIII: Random Forest Confusion Matrix for the 1st Dataset

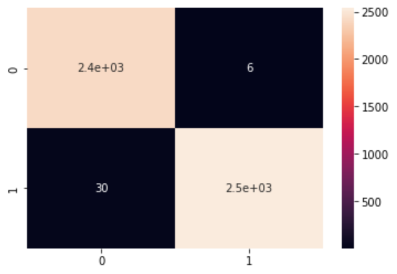


Figure XXXIX: Logistic Regression Confusion Matrix for the 2nd Dataset

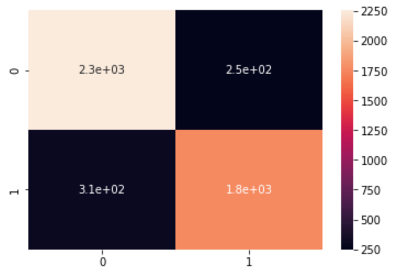


Figure XL: Decision Tree Confusion Matrix for the 2nd Dataset

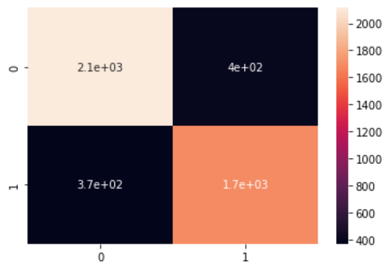


Figure XLI: Naïve Bayes Confusion Matrix for the 2nd Dataset

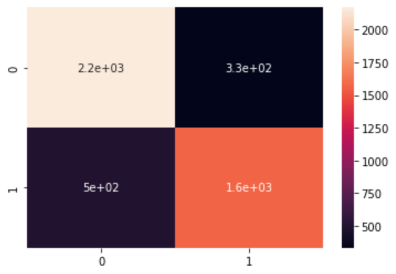
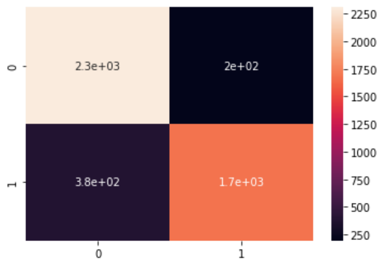


Figure XLII: Random Forest Confusion Matrix for the 2nd Dataset



REFERENCES

[1] C. Bisaillon, “Fake and real news dataset,” *Kaggle*, 26-Mar-2020. Available: https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset.

[2] “Fake news,” *Kaggle*. Available: https://www.kaggle.com/competitions/fakenews/data?select=submit.csv.

[3] C. Beveridge, “33 twitter statistics that matter to marketers in 2022,” *Social Media Marketing & Management Dashboard*, 16-Mar-2022. Available: https://blog.hootsuite.com/twitter-statistics/#:~:text=31.,200%20billion%20Tweets%20per%20year.

[4] “How facebook uses super-efficient AI models to detect hate speech,” *Meta AI*, 19-Nov-2020. Available: https://ai.facebook.com/blog/how-facebook-uses-super-efficient-ai-models-to-detect-hate-speech/.

[5] E. Thompson, “Poll finds 90% of Canadians have fallen for fake news | CBC News,” *CBCnews*, 11-Jun-2019. Available: https://www.cbc.ca/news/politics/fake-news-facebook-twitter-poll-1.5169916.

[6] K. Garneau, C. Zossou, and N. Minnema, “Misinformation during the COVID-19 pandemic,” *Government of Canada, Statistics Canada*, 02-Feb-2021. Available: https://www150.statcan.gc.ca/n1/pub/45-28-0001/2021001/article/00003-eng.htm.

[7] X. Jose, S. D. M. Kumar and P. Chandran, "Characterization, Classification and Detection of Fake News in Online Social Media Networks," 2021 IEEE Mysore Sub Section International Conference (MysuruCon), 2021, pp. 759-765, doi: 10.1109/MysuruCon52639.2021.9641517.

[8] R. Garg and J. S, "Effective Fake News Classifier and its Applications to COVID-19," 2021 IEEE Bombay Section Signature Conference (IBSSC), 2021, pp. 1-6, doi: 10.1109/IBSSC53889.2021.9673448.

[9] S. Zhang, Y. Wang and C. Tan, "Research on Text Classification for Identifying Fake News," 2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), 2018, pp. 178-181, doi: 10.1109/SPAC46244.2018.8965536.

[10] B. Ganesh and D. K. Anitha, "Implementation of Personality Detection and Accuracy Prediction for identification of fake and true news using Decision Tree and Random Forest Algorithms," 2022 International Conference on Business Analytics for Technology and Security (ICBATS), 2022, pp. 1-5, doi: 10.1109/ICBATS54253.2022.9759039.

[11] “Sklearn.linear_model.logisticregression,” *scikit*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[12] H. Yadav, “Logistic regression implementation in Python,” *Medium*, 03-Jul-2021. Available: https://medium.com/machine-learning-with-python/logistic-regression-implementation-in-python-74321fafa95c.

[13] P. Gupta, “Decision trees in machine learning,” *Medium*, 12-Nov-2017. Available: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

[14] “Decision Trees,” *scikit*. Available: https://scikit-learn.org/stable/modules/tree.html#tree.

[15] Saed Sayad, *Naive bayesian*. Available: https://www.saedsayad.com/naive_bayesian.htm.

[16] “Naive Bayes,” *scikit*. Available: https://scikit-learn.org/stable/modules/naive_bayes.html.

[17] T. Yiu, “Understanding random forest,” *Medium*, 29-Sep-2021. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[18] “Ensemble Methods,” *scikit*. Available: https://scikit-learn.org/stable/modules/ensemble.html#forest.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional Transformers for language understanding,” *arXiv.org*, 24-May-2019. Available: https://arxiv.org/abs/1810.04805.

[20] Saumyab271, “Text classification using Bert and tensorflow,” *Analytics Vidhya*, 31-Dec-2021. Available: https://www.analyticsvidhya.com/blog/2021/12/text-classification-using-bert-and-tensorflow/.