UNMASKING DECEPTION: FAKE DETECTION USING NEWS ARTICLE ANAYSIS

Capstone Final Report

MS in Business Analytics

Kent State University

By,

GROUP-5

Abinaya Sundari Panneerselvam

Harish Kumar Uddandi

Bharath Bhimireddy

Advisor:

Dr. Dambar Uprety

Asst Professor

Dept of Information System & Business Analytics

Kent State University

Table Of Contents:

List Of Figures

UNMASKING DECEPTION: FAKE DETECTION USING NEWS ARTICLE ANAYSIS

1. Introduction

About the Dataset

The dataset consists of news articles associated with attributes that can be leveraged for analyzing and detecting potentially unreliable information, commonly called "fake news." This dataset is crucial in exploring "Unmasking Deception: Fake Detection using News Article Analysis."

The training dataset, named "train.csv," contains the following attributes:

Id: A unique identifier assigned to each news article, facilitating easy referencing and organization.

Title: The news article's title often provides a concise summary of the article's content and subject matter.

Author: The name of the author who penned the news article, which can be valuable for understanding potential biases or perspectives.

Text: The article's main body encompasses the core information and context. The text may occasionally need to be completed, making analysis more challenging.

Label: A categorical label assigned to each article, indicating whether it is deemed potentially unreliable (1) or reliable (0). These labels are the ground truth for training a fake news detection model.

Additionally, the testing dataset, "test.csv," shares the same attributes as the training dataset, except for the absence of the "label" attribute. This dataset is used to evaluate the performance of the trained model on unseen data.

The main goal of this project is to create a strong machine learning model that can effectively identify fake news by using the text content and related metadata given in the dataset. The model will undergo training using the labeled training data, and its effectiveness will be assessed through its predictions on the testing data.

As digital media becomes more prevalent, it's crucial to identify trustworthy information and combat the spread of false content. This dataset and its analysis are vital tools in promoting media literacy and countering deception. By unmasking deception and developing accurate fake news detection mechanisms, we can take significant steps toward fostering a more informed and trustworthy information ecosystem.

2. Project Scope and Objective:

False news is intentionally made up in a manner that looks authentic and aims to influence people's view of actual events. Inaccurate and deceptive content takes on various modes of communication, such as written, spoken, and digital/electronic forms. Importantly, fake news or disinformation is a deliberate form of deception that capitalizes on the intent to manipulate the reader or audience. This notion separates itself from misinformation, created regardless of the intent to deceive. When individuals or groups knowingly share or re-post misinformation to deliberately cast doubt on an event, misinformation can easily become disinformation. The problem of "fake news" has been a persistent theme in the headlines for decades, but most notably within the past few years.

Scenario: Exposing a Politically Motivated Fake News Campaign

In politics, fake news can be a powerful tool to sway public opinion and manipulate electoral outcomes. Let's consider a scenario to illustrate the significance of our research on "Unmasking Deception: Fake Detection Using News Article Analysis."

Example: The "Foreign Interference" Campaign

During a critical election season in a democratic country, a coordinated fake news campaign emerges to influence the voters' perception of a specific political candidate. The campaign involves creating and disseminating deceptive news articles, opinion pieces, and social media posts. The fake news campaign is fueled by actors with vested interests in seeing a particular candidate succeed or fail. Foreign adversaries seeking to disrupt the country's democratic process are suspected to be behind this operation. They aim to sow seeds of discord, create confusion, and undermine public trust in the electoral system. The fake news articles are carefully crafted to appear as legitimate news reports from reputable media outlets. They contain false allegations, fabricated quotes, and manipulated statistics to tarnish the candidate's image and reputation. Fake news gaining traction on social media and other online platforms influences public opinion. Many citizens question the candidate's credibility, causing divisions among the electorate and undermining the democratic process.

Our research on "Unmasking Deception: Fake Detection Using News Article Analysis" becomes crucial in this scenario. By leveraging advanced natural language processing (NLP) algorithms and machine learning techniques, our fake news detection system can analyze news articles' content, identify deception patterns, and flag suspicious articles for further review.

Our system can expose the fake news campaign's origin and motives through sentiment analysis, linguistic cues, and cross-referencing with reputable sources. It can reveal the links between the deceptive articles and the foreign actors seeking to disrupt the democratic process.

With this information, election officials, media organizations, and the public can take proactive measures to combat the spread of fake news. Fact-checking initiatives and media literacy campaigns can be intensified to educate the public about deceptive content and how to identify trustworthy sources of information.

Ultimately, the successful detection and exposure of the fake news campaign can help safeguard the integrity of the democratic process. By unmasking deception and promoting accurate information, our research contributes to strengthening the resilience of democratic societies against the manipulation tactics of fake news purveyors.

3. Literature Review:

Article 1: Ahmed H, Traore I, Saad S. "Detecting opinion spams and fake news using text classification," Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.

The literature review discusses a research paper focused on detecting fake content, including fake news and fake reviews, using text classification methods. Deceptive content has become a significant concern in recent years, impacting consumers and businesses. The paper proposes a novel n-gram model combined with text analysis and machine learning classifiers to detect fake content automatically. The study categorizes fake reviews into three groups and fake news into three groups to better understand the challenges associated with each type.

Existing works related to fake content detection are explored, which fall into content-based and reviewer behavior-based detection models. Content-based models use text analytics and NLP techniques, while reviewer behavior-based models identify patterns of spammers targeting specific products or brands.

The research paper is regarded as a significant contribution to the field, as it is the first to propose a unified approach capable of detecting both fake news and fake reviews. The model outperforms existing methods when evaluated on various datasets, including a newly collected fake news dataset.

For future work, the authors prioritize incorporating statistical and writer-style features into the model, such as slang word count and filler word usage. They also suggest exploring feature selection techniques to handle the high dimensionality of text-based features.

The paper offers valuable insights on detecting fake content and presents a unified and effective approach to aid researchers in this growing field, highlighting challenges and advancements.

Article 2: Mukherjee A, Venkataraman V, Liu B, Glance N. Fake review detection: classification and analysis of real and pseudo reviews. UIC-CS-03-2013. Technical Report; 2013.

The literature on fake review detection has gained significant attention as online platforms strive to maintain the authenticity of user experiences and opinions. Previous research focused on using supervised learning with various features to detect fake reviews, but the lack of reliable labeled data posed challenges.

A breakthrough came with using Amazon Mechanical Turk (AMT) to generate pseudo-fake reviews, achieving high accuracy with word bigram features. However, the AMT-generated reviews were not representative of real fake reviews found on commercial websites.

A comparison was made between AMT data and real-life Yelp reviews to address this. Detecting fake reviews in a real-life setting proved harder. An innovative method using KL-divergence revealed differences between AMT-generated fake reviews and real-life Yelp reviews, highlighting the linguistic pretense of Yelp fake reviewers.

Behavioral features about reviewers and their reviews were proposed to improve real-life fake review detection, significantly boosting accuracy. Interestingly, behavioral features alone outperformed n-gram features.

In conclusion, this literature review showcases the progression of fake review detection research and the unique contributions of the current study, which explores AMT-generated fake reviews and real-life Yelp data and introduces behavioral features for enhanced detection.

Article 3: Shojaee S, Murad MAA, Azman AB, Sharef NM, Nadali S. Detecting deceptive reviews using lexical and syntactic features. Paper presented at Intelligent Systems Design and Applications (ISDA), 2013 13th International Conference; December, 2013:53-58; Seri

The introduction of the research paper discusses the importance of social media as a source of information about users' behaviors and interests. While this information can be beneficial for companies and governments to understand public opinions, the widespread use of user-contributed reviews has raised concerns about their reliability. The increasing number of reviews makes manual detection and examination of fake reviews impractical due to information overload.

The paper acknowledges that previous works have focused on spotting review spam that is easily detectable by humans. Still, the concern now lies in detecting deceptive reviews that are more challenging to identify. To address this, the paper proposes using stylometric (writing style) features to classify deceptive and trustworthy reviews. In this study, stylometric features are classified into lexical and syntactic. The authors use two classifiers, Support Vector Machine (SVM) with Sequential Minimal Optimization (SMO) and Naive Bayes, to classify deceptive reviews. They summarize previous research, introduce the deceptive opinion spam corpus used in the study, and explain the features. The experimental section presents the results of applying the classifiers to different feature types. Finally, the conclusion examines the effectiveness of stylometric features for detecting deception. It suggests future research directions, such as extracting content-specific features and investigating the combination of features to achieve higher accuracy.

The paper aims to show that utilizing stylometric features is a hopeful method for identifying fraudulent reviews. It also offers valuable guidance on using language cues to differentiate between honest and deceitful reviews. The experimental results support the effectiveness of stylometric features in this context. Still, the authors acknowledge that further research is needed to explore alternative techniques and optimize feature sets for improved performance.

Article 4: Sami Ben Jabeur, Hossein Ballouk, Wissal Ben Arfi, Jean-Michel Sahut, Artificial intelligence applications in fake review detection: Bibliometric analysis and future avenues for research, Journal of Business Research, 10.1016/j.jbusres.2022.113631, 158, (113631), (2023)

Fake review detection using artificial intelligence (AI) and machine learning (ML) techniques has become an essential research area, considering the significant impact of fake reviews on online commerce. The literature has observed cases where a vast majority of reviewers write only one review (singleton review), influencing store ratings and impressions. However, existing methods need to pay more attention to these reviewers. This review proposes a novel approach that considers the temporal patterns of reviewers' behavior to detect fake reviews.

Previous research has primarily focused on three main groups of fake review detection: (a) word of mouth, quality, reputation, and price; (b) classification, moderating role, intention, and analytics; and (c) impact and participation. Traditional methods have been augmented with psychology and computational linguistics to improve detection accuracy. Methods such as classifier compilation, binomial regression, and review graph analysis have been proposed to address the problem.

Recent works have explored the application of AI, ML, and deep learning (DL) techniques to detect fake reviews. N-gram models, Latent Dirichlet Allocation (LDA)-based computing frameworks, and time-series data analysis have been utilized to uncover suspicious patterns in fake reviews.

A hybrid integrated review combining bibliometric analysis with a framework-based review was conducted to gain a comprehensive understanding of AI applications in fake review detection. The 4Ws (What, Where, Why, and How) were employed to provide a better structure for this review, aiding in identifying research gaps and potential future research directions.

Overall, this literature review highlights the growing interest in using AI and ML techniques for fake review detection. It provides valuable insights into the advancements and hotspots in the field and emphasizes the importance of further research in effectively detecting fraudulent reviews using advanced AI techniques.

Article 5: A. K. Dutta, B. Qureshi, Y. Albagory, M. Alsanea, M. A. Faraj, et al., "Optimal weighted extreme learning machine for cybersecurity fake news classification," Computer Systems Science and Engineering, vol. 44, no.3, pp. 2395–2409, 2023.

Fake news has become a significant concern, affecting various aspects of society, and creating challenges distinguishing between real and fake information. Experts have developed machine learning models to address the issue of fake news by identifying and categorizing such content. This review paper proposes a novel approach called Chaotic Ant Swarm with Weighted Extreme Learning Machine (CAS-WELM) for Cybersecurity Fake News Detection and Classification.

The review highlights the prevalence of dramatic headlines and clickbait titles that contribute to the spread of inaccurate and unprofessional news for the sake of advertising revenue. The dissemination of fake news can lead to political manipulation, financial gains, and various damages. Detecting fake news is challenging, and AI and NLP (natural language processing) techniques have shown potential in combating the spread of false information.

Previous research in fake news detection has employed various ML methods, including attention mechanisms, CNN, LSTM, and ensemble approaches. Word embedding techniques like GloVe have also been utilized to establish linguistic relationships and create word vectors. The paper presents several DL-based algorithms for classifying fake news, demonstrating their effectiveness in differentiating between real and fake information.

The CAS-WELM technique proposed in this study aims to classify news into fake and real categories. It involves preprocessing the input data, employing GloVe for word embedding, using N-gram-based feature extraction, applying WELM for classification, and optimizing the weight values using the CAS algorithm. The experiment results show that CAS-WELM performs better than other recent methods.

To sum up, the paper highlights the significance of using advanced ML and DL techniques to tackle the issue of fake news on social networking platforms. The proposed CAS-WELM technique shows promising results in detecting and classifying fake news, contributing to the ongoing efforts to identify and mitigate the spread of false information.

4. Data and Variables:

Dataset Description:

Data Source: Kaggle

1. Train.csv:

A complete training set having the following characteristics:

▪ id: unique identifier for a news item.

▪ title: News article's title

▪ author: News article's author

▪ There are two columns in this data: Text, Label

▪ Text: The article's text might be missing some information.

▪ Label: A designation that indicates the article might not be

reliable.0: Reliable, 1: Unreliable

2. Test.csv: A training dataset for testing that has all the same characteristics as

train.csv but without the label.

5. Model and Method:

Data Loading and Preprocessing:

The main objective is to perform preprocessing and feature extraction using the Doc2Vec model for fake news detection. Let's break down the main steps and understand what is happening in the code:

The code reads the training data from the CSV file 'datasets/train.csv' into a Panda DataFrame.

An overview of the objectives and methods used in the report, focusing on fake news detection using Doc2Vec preprocessing.

Cleaning Text (Step 1): textClean function

The textile function removes non-letter and non-number characters from the text and filters out stopwords using the NLTK library.

```
def textClean(text):
    """
    Get rid of the non-letter and non-number characters.
    """
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
    text = text.lower().split()
    stops = set(stopwords.words("English"))
    text = [W for W in text if not W in stops]
    text = " ".join(text)
    return (text)
```

The cleanup function further cleans the preprocessed text by removing punctuation.

```
def cleanup(text):
    text = textClean(text)
    text = text.translate(str.maketrans("", "", string.punctuation))
    return text
```

Constructing Labeled Sentences: constructLabeledSentences Function

Labeled sentences are essential for training the Doc2Vec model. The constructLabeledSentences function creates labeled sentences for each news article's preprocessed text.

```
def constructLabeledSentences(data):
    sentences = []
for index, row in data.iteritems():
        sentences.append(LabeledSentence(utils.to_unicode(row).split(), ['Text' + '_%s' %
str(index)]))
    return sentences
```

Generating Doc2Vec Embeddings: getEmbeddings function.

The getEmbeddings function reads the data from the "datasets/train.csv" file, performs data cleaning, and generates Doc2Vec embeddings.

```
def getEmbeddings(path, vector_dimension=300):
    """
    Generate Doc2Vec training and testing data.
    """
    data = pd.read_csv(path)
    # Data Preprocessing
    # Construct labeled sentences
    x = constructLabeledSentences(data['text'])
    y = data['label'].values
    # Doc2Vec Model Training
    text_model = Doc2Vec(min_count=1, window=5, vector_size=vector_dimension,
sample=1e-4,
     negative=5, workers=7, epochs=10,seed=1)
    text_model.build_vocab(x)
    text_model.train(x, total_examples=text_model.corpus_count,
epochs=text_model.epochs)
    # Data Splitting
    return text_train_arrays, text_test_arrays, train_labels, test_labels
```

The clean_data function shuffles the data to randomize the order of training samples, saves the shuffled data, and labels them as NumPy arrays for further use.

```
def clean_data():
    """
    Generate processed string
    """
    path = 'datasets/train.csv'
    vector_dimension = 300
```

```
data = pd.read_csv(path)
# Data Preprocessing
# Data Shuffling
data = data.sample(frac=1).reset_index(drop=True)
x = data.loc[:, 'text'].values
y = data.loc[:, 'label'].values
train_size = int(0.8 * len(y))
test_size = len(x) - train_size
xtr = x[:train_size]
xte = x[train_size:]
ytr = y[:train_size]
yte = y[train_size:]
# Save the processed data as NumPy arrays
np.save('xtr_shuffled.npy', xtr)
np.save('xte_shuffled.npy', xte)
np.save('ytr_shuffled.npy', ytr)
np.save('yte_shuffled.npy', yte)
```

The code performs the entire Doc2Vec preprocessing pipeline with these functions, including text cleaning, constructing labeled sentences, generating Doc2Vec embeddings, and shuffling and saving the processed data. The saved data is now ready for use in model training and evaluation for fake news detection.

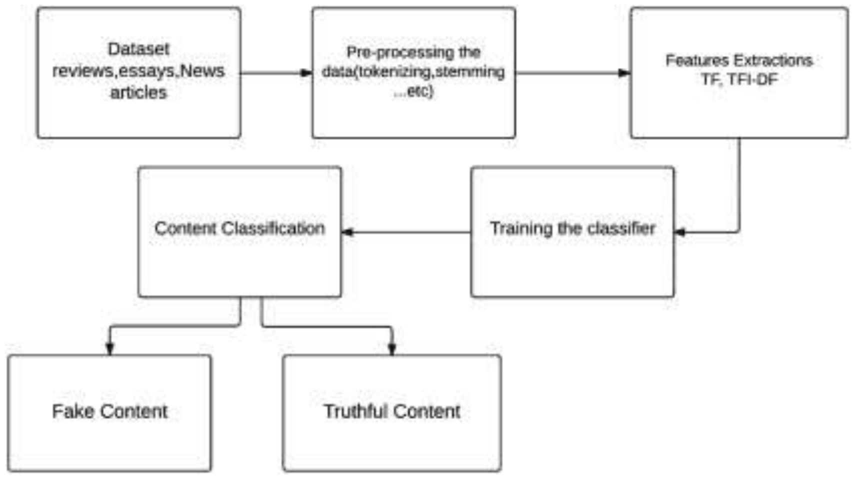

Dataset reviews,essays,News articles
Pre-processing the data(tokenizing,stemming ...etc)
Features Extractions TF, TFI-DF
Training the classifier
Content Classification
Fake Content
Truthful Content

Fig 1: Label Distribution

5.1. Exploratory Data Analysis (EDA):

Before modeling, we can inspect what our data looks like. We cannot assume that our raw data is completely clean and ready for modeling. Instead, we must convert some variables to factors, remove missing values, and clean our predictor names. Text-cleaning manipulation will also be performed on our data for our natural language preprocessing step. Focus on developing and evaluating models for accurately detecting fake news. We preprocess the data by handling missing values and cleaning the text.

Our models include Naive Bayes, SVM, Neural Networks (using TensorFlow and Keras), and LSTM. Each model has its strengths and weaknesses, such as efficiency, handling high-dimensional data, and capturing complex patterns. We evaluate their performance using accuracy, precision, recall, and F1 score metrics. The LSTM model performs well with the highest accuracy. However, further analysis is recommended, including hyperparameter tuning, cross-validation, feature importance analysis, and addressing class imbalance.

The goal is to contribute to maintaining information integrity in the digital age by developing reliable systems for fake news detection.

5.2. Target Variable and Data Preprocessing:

The target variable for the fake news detection project is called "label," representing the binary classification of news articles as either fake or genuine.

To prepare the data for modeling, several preprocessing steps are applied. Missing values in the text data are handled differently for different models: Naive Bayes, SVM, and Neural Network models:

Rows in the dataset where the "text" column is missing, or NaN are dropped using the drop () Function from the panda's library.

After dropping the missing rows, the remaining data is used for training and testing the models. Doc2Vec preprocessing: Missing values in the "text" column are identified using the condition data.loc[i, 'text'] != data.loc[i, 'text'].

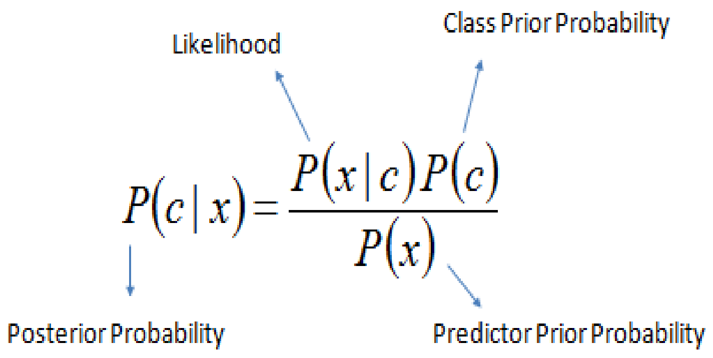
Rows with missing values are dropped using the drop() Function from the pandas' library. After removing the missing rows, the remaining data is cleaned and preprocessed.

5.3. Identifying Possible Models:

For the fake news detection project, several models can be employed. It is crucial to consider both traditional statistical models and machine learning models. Some commonly used models for this task include:

5.3.1 Naive Bayes:

Naive Bayes is a probabilistic model that assumes independence between features given the class label. It works well with text-based data, handles high-dimensional feature spaces efficiently, and provides fast predictions. However, it may not capture complex relationships between features and can be affected if the independence assumption is violated.


Likelihood
Class Prior Probability
$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

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

Fig 2: Naïve Bayes

5.3.2 SVM (Support Vector Machine):

SVM is a powerful model for classification tasks. It aims to find an optimal hyperplane that separates different classes or approximates a regression function. SVMs handle high-dimensional data, both linear and non-linear relationships, and class imbalance well. Sometimes, these can be quite costly in terms of computation and need a careful choice of hyperparameters.

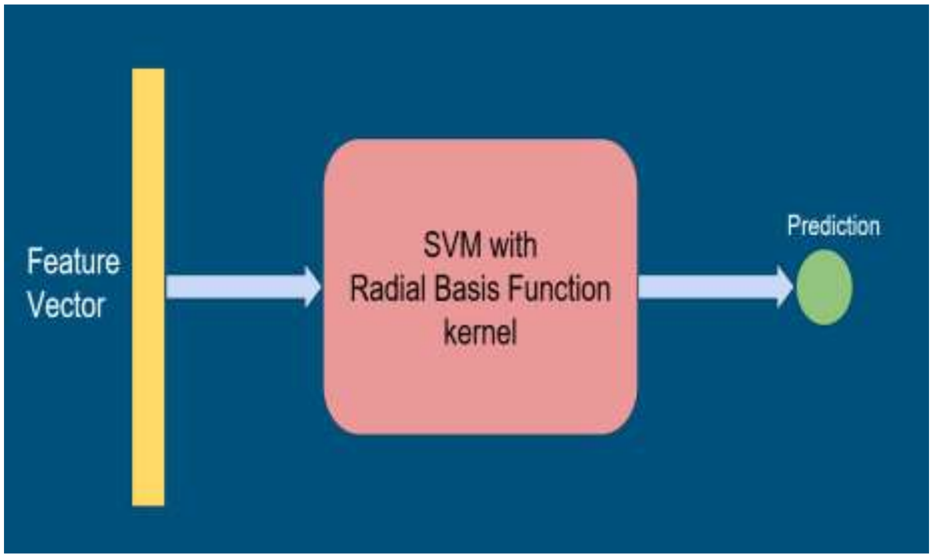


Fig 3: Support Vector Machine

5.3.3 Neural Network:

Neural networks, particularly deep learning models, have gained popularity in various domains, including fake news detection. They can capture complex patterns and relationships in data, handle high-dimensional data, and excel at capturing non-linear relationships. However, they require a large amount of labeled training data and computational resources for training. Architecture design, hyperparameter tuning, and regularization techniques are important for preventing overfitting.

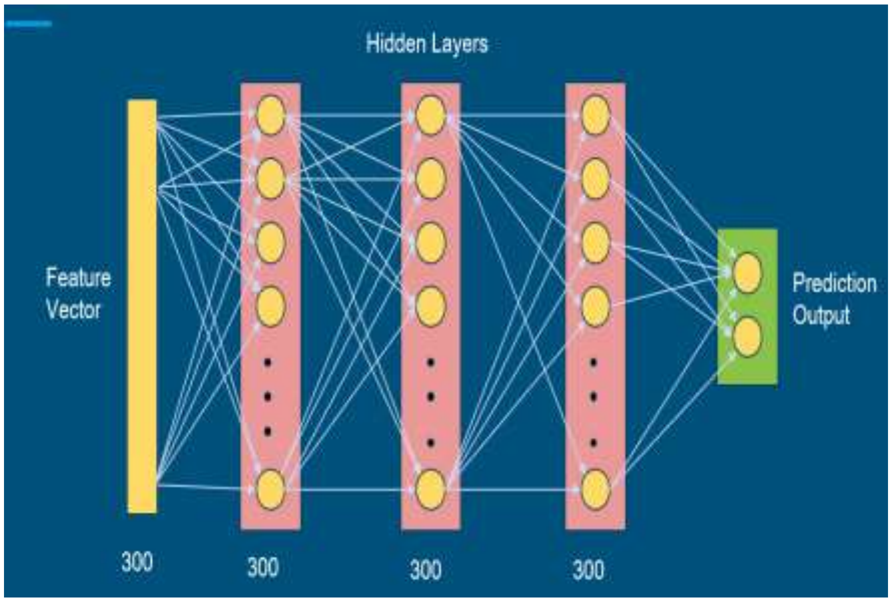


Fig 4: Neural Network

5.3.4 LSTM (Long Short-Term Memory):

Recurrent neural networks (RNNs) of the LSTM type are created especially for handling sequential data, such as text or time series. LSTMs effectively capture long-term dependencies and patterns, making them suitable for text analysis tasks. However, they can be computationally expensive, require careful hyperparameter tuning, and may suffer from overfitting.

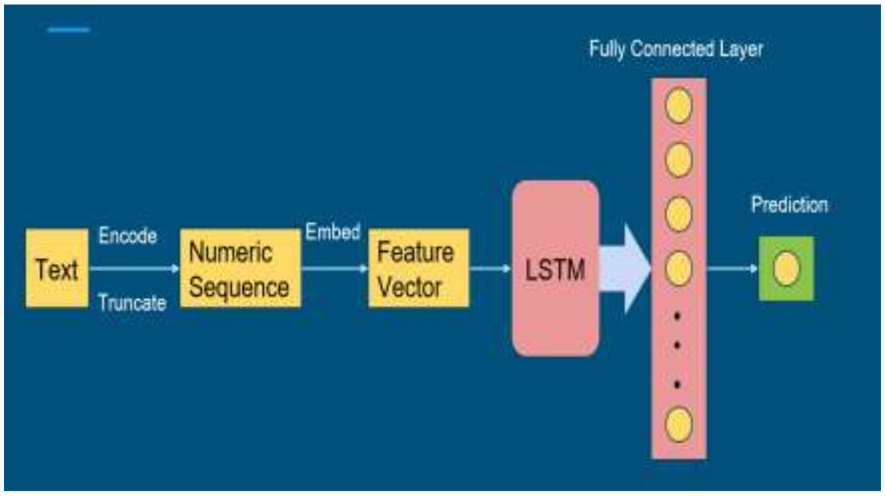


Fig 5: LSTM (Long Short-Term Memory)

5.4. Model Processing Order:

The models are processed in the following order:

5.4.1 Naive Bayes Model:

The Naive Bayes model is processed first. It is trained and evaluated using the GaussianNB classifier from the sci-kit-learn library. The model's accuracy is calculated, and the confusion matrix is plotted.

5.4.2 LSTM Model:

The Long Short-Term Memory (LSTM) model is processed in seconds. It is built using the Keras library with embedding and LSTM layers. The model is trained and evaluated, and the accuracy and confusion matrix is calculated and plotted.

5.4.3 Neural Network Model (Keras):

The Neural Network model implemented using the Keras library is processed next. It consists of multiple dense layers with dropout regularization. The model is trained and evaluated, and the accuracy and confusion matrix is calculated and plotted.

5.4.4 SVM Model:

The Support Vector Machine (SVM) model is processed last. It is trained and evaluated using the SVC classifier from the sci-kit-learn library. The model's accuracy is calculated, and the confusion matrix is plotted.

5.5 Dataset Characteristics and Model-Data Alignment:

The dataset contains a CSV file with the "text" column representing the textual data of news articles and the "label" column containing binary labels indicating fake or genuine news. The preprocessing steps in the code align with the assumed dataset characteristics. The code handles missing values in the "text" column, performs text cleaning and preprocessing, and prepares the data for training and testing the models.

5.6 Data Loading and Preprocessing:

The code reads the dataset from the provided path using pd.read_csv. It then checks for missing rows in the 'text' column and removes them. The cleaning process involves removing non-alphanumeric characters, converting text to lowercase, removing stop words, and returning the words to a string.

5.7 Constructing Labeled Sentences:

The constructLabeledSentences function takes the preprocessed text data and creates labeled sentences using the labeled sentence class from Gensim. Each sentence is assigned a label in the format 'Text_i,' where I represent the index of the sentence in the dataset.

5.8 Generating Doc2Vec Embeddings:

The get Embeddings function takes the preprocessed data, constructs labeled sentences, and trains a Doc2Vec model using Gensim. The model is then used to generate vector representations (embeddings) for each labeled sentence.

The function splits the data into training and testing sets, converts the labeled sentence embeddings into NumPy arrays, and returns the training and testing data along with the corresponding labels.

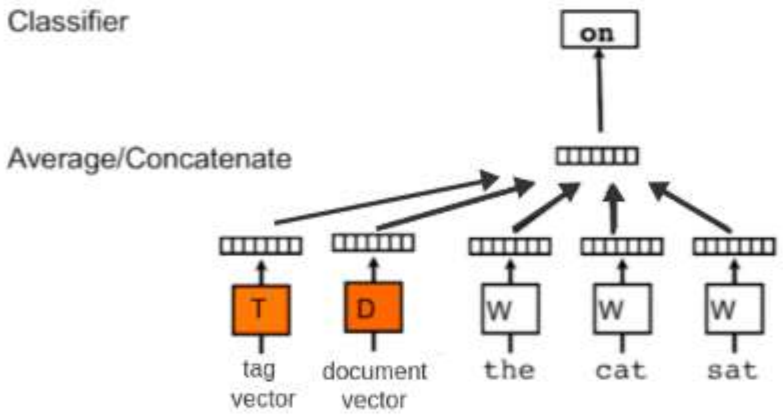


Fig 6: Doc2Vec Model

5.9 Cleaning Data:

It shuffles the dataset and splits it into training and testing sets. The preprocessed data and labels are then saved into individual NumPy files.

Regarding the dataset characteristics, the code assumes the presence of a CSV file at the specified path, containing 'text' and 'label' columns. The 'text' column represents the textual data of each news article, and the 'label' column contains the corresponding binary labels indicating fake or genuine news. The code performs text cleaning, shuffling, and splitting of the data into training and testing sets.

It generates Doc2Vec embeddings for preprocessed text data.

5.9.1 Naive Bayes model:

Data Loading and Preprocessing: The code snippet checks if the necessary data files (xtr.npy, xte.npy, ytr.npy, yte.npy) exist. If they don't, it calls the getEmbeddings function to read the data from the "datasets/train.csv" file and saves the processed data into the respective numpy files.

However, the code does not include implementing the getEmbeddings function, which might be responsible for data preprocessing and feature extraction. You need to provide the code or function to complete this step.

Model Training and Prediction: After loading the preprocessed data, the code trains a Naive Bayes classifier (GaussianNB) using the xtr and ytr data. Then, it predicts the target variable (y_pred) for the test data (tx) using the trained model. Model Evaluation:

The code calculates the accuracy of the model by comparing the predicted labels (y_pred) with the true labels (yte). The number of incorrect predictions is counted, and the accuracy is calculated as a percentage.

Confusion Matrix: The code calls the plot_cmat function to visualize the confusion matrix, which shows the distribution of predicted labels against true labels.

5.9.2 The LSTM model for fake news detection:

Data Loading and Preprocessing: The code checks if the necessary preprocessed data files (xtr_shuffled.npy, xte_shuffled.npy, ytr_shuffled.npy, yte_shuffled.npy) exist. If they don't, it calls the clean_data function from the getEmbeddings module to perform data cleaning and preprocessing.

I. **Data Encoding and Transformation**: The code initializes a Counter object to count the frequency of words in the dataset. It then processes the training data by splitting the news into words, updating the Counter, and replacing words with their corresponding IDs from the word bank dictionary. The same encoding and transformation process is applied to the test data.
II. **Data Truncation and Padding:** To ensure uniform input sequence length, the code truncates or pads the encoded sequences to a specified length (max_review_length). This step is necessary as LSTMs require input sequences of equal length. The sequence.pad_sequences function from Keras is used for this purpose.
III. **Model Creation and Training:** The code constructs an LSTM-based model using Keras Sequential API. The model consists of an embedding layer, an LSTM layer, and a dense layer with sigmoid activation. It then compiles the model using binary cross-entropy loss and the Adam optimizer. During training, the model uses X_train and y_train as the training data and receives validation data.
IV. **Model Evaluation:** After training, the code evaluates the model's performance on the test data (X_test and y_test) and prints the accuracy score.
    Confusion Matrix: The code calls the plot_cmat function to visualize the confusion matrix, which provides insights into the distribution of predicted labels against true labels. Regarding the dataset characteristics, the code assumes the presence of preprocessed data files and does not provide specific information about the dataset's nature, size, or target variable. However, based on the code, we can infer that the dataset consists of news articles, each represented by a sequence of words.
    The variable we're targeting has only two possible values: 0 or 1. These values represent fake news and genuine news, respectively. We encode words as numerical IDs to work with the text data and then use an embedding layer to represent each word as a dense vector.

5.9.3 Neural Networks model for fake news detection (Keras, TensorFlow):

I. Implements a Keras neural network model for fake news detection. Data Loading and Preprocessing:

The code checks if the necessary data files (xtr.npy, xte.npy, ytr.npy, yte.npy) exist. If they don't, it calls the getEmbeddings function to read and preprocess the data from the "datasets/train.csv" file. The preprocessed data is then saved into the respective numpy files. Model Architecture: The code defines a neural network model with three hidden layers using the Keras Sequential API.

The model architecture consists of dense layers with dropout regularization. The last layer has two units with softmax activation to predict the binary classes (fake or genuine news). The model is compiled with categorical cross-entropy loss and stochastic gradient descent (SGD) optimizer.

Model Training and Evaluation: The model is trained using the training data (x_train and y_train) and evaluated using the testing data (x_test and y_test). The train_test_split function divides the data into training and testing sets. The target variable is encoded using the one-hot encoding (np_utils.to_categorical) to match the model's output format. 64 batches of 20 epochs each are used to train the model. The accuracy of the model is then evaluated and printed.

Confusion Matrix: The code uses the plot_cmat function to visualize the confusion matrix, which provides insights into the distribution of predicted labels against true labels. Regarding the dataset characteristics,

the code assumes the presence of the preprocessed data files (xtr.npy, xte.npy, ytr.npy, yte.npy). The dataset is expected to have numerical features of shape (number of samples, number of features) in the xtr and the files. The target variable (ytr and yte) should be binary labels indicating fake or genuine news. The code handles the encoding of the target variable and trains the neural network model accordingly.

II. Implements a TensorFlow neural network model for fake news detection. Data Loading and Preprocessing:

The code checks if the necessary data files (xtr.npy, xte.npy, ytr.npy, yte.npy) exist. If they don't, it calls the getEmbeddings function to read and preprocess the data from the "datasets/train.csv" file. The preprocessed data is then saved into the respective NumPy files.

Model Function: The model_fn Function defines the TensorFlow model architecture using the tf.estimator API. It consists of several dense layers with dropout regularization. The final layer has two units corresponding to the two classes (fake or genuine news).

The model supports the PREDICT, TRAIN, and EVAL modes. Training and Evaluation: The code creates an Estimator object using the defined model function. It sets up logging hooks for prediction and trains the model using the training data. The model is evaluated using the evaluation data, and the evaluation results are printed.

Confusion Matrix: The code uses the plot_cmat function to visualize the confusion matrix, which provides insights into the distribution of predicted labels against true labels. Regarding the dataset characteristics, the code assumes the presence of the preprocessed data files (xtr.npy, xte.npy, ytr.npy, yte.npy). The dataset should contain numerical features in the xtr and the files representing the Doc2Vec embeddings.

The target variable should be binary labels indicating fake or genuine news present in the ytr and yte files. The code handles the input and output shapes accordingly and trains the TensorFlow model.

5.9.4 Support Vector Machine (SVM) model for fake news detection:

Data Loading and Preprocessing: The code checks if the necessary data files (xtr.npy, xte.npy, ytr.npy, yte.npy) exist. If they don't, it calls the getEmbeddings function to read and preprocess the data from the "datasets/train.csv" file. The preprocessed data is then saved into the respective NumPy files.

Model Training and Evaluation: The code uses the built-in SVC (Support Vector Classification) from Scikit-learn to train the SVM model. It fits the model using the training data (xtr and ytr). Once the model has been trained, it can forecast the labels for the testing data (xte). The accuracy is determined by contrasting the predicted labels (y_pred) against the actual ground truth labels (yte). Finally, the accuracy is displayed.

Confusion Matrix: The code uses the plot_cmat function to visualize the confusion matrix, providing insights into the distribution of predicted labels against true labels. Regarding the dataset characteristics, the code assumes the presence of the preprocessed data files (xtr.npy, xte.npy, ytr.npy, yte.npy).

The dataset should contain numerical features in the xtr and the files. The target variable should be binary labels indicating fake or genuine news present in the ytr and yte files. The code trains the SVM model using the training data and evaluates its accuracy using the testing data.

5.10. Performance Metrics:

To evaluate the performance of the models, it is essential to define the most relevant performance metrics for the project. For classification problems like fake news detection, metrics such as accuracy, precision, recall, and F1 score are commonly used. Calculate the accuracy of each model by comparing the predicted and true labels. Accuracy provides an overall measure of the model's performance. A confusion matrix is created to understand better the model's performance in predicting labels. This matrix helps visualize the distribution of predicted labels against the true labels. It also provides insights into the number of false positives, false negatives, true positives, and true negatives.

While it is the lowest among the models evaluated, it still performs significantly better than random guessing. Naive Bayes is known for its computational efficiency and ability to handle high-dimensional data, making it a viable option for fake news detection tasks with limited computational resources.

SVM: The SVM model achieved an accuracy of 88.42%, which is a notable improvement compared to Naive Bayes. SVMs are known for handling high-dimensional data, capturing linear and non-linear relationships, and handling class imbalance well. With its strong performance, SVMs can be considered a reliable option for fake news detection.

Neural Network with TF: The Neural Network model implemented with TensorFlow achieved an accuracy of 81.42%. Neural networks, including TensorFlow-based models, can capture complex non-linear relationships in the data. While the accuracy achieved by this model is lower compared to SVM and LSTM, it still demonstrates promise in identifying fake news.

Neural Network with Keras: The Neural Network model implemented with Keras achieved an accuracy of 92.62%. Keras-based neural networks excel in handling high-dimensional data, adaptability to various problem domains, and capturing non-linear relationships. With a high accuracy rate, this model shows strong potential for fake news detection.

LSTM: The Long Short-Term Memory (LSTM) model achieved the highest accuracy among the models evaluated, with a rate of 94.53%. LSTMs are specifically designed for handling sequential data like text and have proven successful in natural language processing tasks. With its ability to capture long-term dependencies and context, the LSTM model is well-suited for fake news detection.

Based on the accuracy results, it can be concluded that the LSTM model performed the best, followed by the Neural Network with Keras. These models demonstrated the highest accuracy rates in detecting fake news. SVM also performed well, achieving a high accuracy rate. Naive Bayes and the Neural Network with TensorFlow achieved relatively lower accuracy rates but can still be considered viable options, especially considering their computational efficiency.

Model	Accuracy
Naive Bayes	75.96%
SVM	86.87%
Neural Network with TF	83.58%
Neural Network with Keras	92.68%
LSTM	96.76%

Table 1: Model's Accuracy

5.11. Model Exploration and Further Analysis:

To thoroughly explore the models and make informed decisions, it is recommended to conduct further analysis. Here are some suggestions: Hyperparameter Tuning: Experiment with different hyperparameters for each model to find the optimal configuration. One way to achieve this is through methods such as grid search or random search. Adjusting hyperparameters can have a significant effect on the model's overall performance.

Cross-Validation: Instead of a simple train-test split, consider using cross-validation techniques such as k-fold cross-validation. Evaluating the models on multiple train-test splits helps obtain more reliable performance estimates. Additional Performance Metrics: Explore other performance metrics like precision, recall, and F1 score apart from accuracy. These metrics provide insights into a model's ability to correctly classify fake and genuine news, considering both false positives and false negatives.

Feature Importance: Investigate the importance of different features or words in the models. Techniques like feature importance analysis, word frequency analysis, or word embedding visualization can help identify the most influential features or words in determining the news article's authenticity. Handling Class Imbalance: If the distribution of fake and genuine news in the dataset is skewed, it may be helpful to address class imbalance through techniques such as oversampling, under-sampling, or using advanced algorithms like SMOTE (Synthetic Minority Over-sampling Technique) to create synthetic samples.Model Comparison: Compare the performance of different models using a variety of metrics.

Consider the trade-offs between accuracy, interpretability, complexity, computational requirements, and other relevant factors specific to your project requirements. Ensembling or Stacking: Explore the possibility of combining multiple models through ensembling or stacking techniques. Ensemble models combine the predictions of multiple models to improve overall performance and reduce bias or variance. Stacking involves training a meta-model on the predictions of individual models to capture their collective knowledge.

6. Model validation test:

6.1.1 Naïve Bayes Model:

The "Naive Bayes model" script focuses on employing the Gaussian Naive Bayes classifier for detecting fake news and evaluating its effectiveness. The process involves several steps and functions to accomplish the task.

Firstly, the necessary modules and libraries are imported, and the script also utilizes the "getEmbeddings" method from another script.

The code includes a function called "plot_cmat," which is responsible for plotting the confusion matrix. This matrix helps visualize the model's performance by showing the number of true positive, true negative, false positive, and false negative predictions.

Before proceeding, the script checks for the existence of preprocessed data files such as "ytr.npy," "xtr.npy," and "xte.npy." If these files are unavailable, the script uses the "getEmbeddings" function to obtain the preprocessed data and stores it as numpy files for future use.

Subsequently, the preprocessed data, including training and test datasets and their corresponding labels (xtr, xte, ytr, yte), is loaded from the saved numpy files.

The Gaussian Naive Bayes classifier is then initialized, and the training data (xtr, ytr) is used to train the classifier.

After training the model, predictions are generated based on the test data (tx), and the results are stored in the variable "y_pred."

The script compares the predicted labels (y_pred) with the actual labels (yte) from the test dataset to assess the model's accuracy.

Finally, the "plot_cmat" function visually represents the confusion matrix, clearly understanding the model's performance.

In conclusion, the "Naive Bayes model" script demonstrates the implementation of the Gaussian Naive Bayes classifier for fake news detection. It evaluates its effectiveness through various steps, including data loading, model training, prediction generation, and accuracy assessment. Visualizing the confusion matrix provides valuable insights into the model's performance distinguishing between genuine and fake news articles.

6.1.2 Neural Network Model- Keras:

The "Keras version of neural network" script is dedicated to defining, training, and evaluating the performance of a neural network model for detecting fake news. The functions used in the code contribute to different aspects of this process.

Firstly, the script imports the required modules and libraries, including the "getEmbeddings" method from another script. It also defines the "plot_cmat" function, which helps visualize the confusion matrix.

To ensure the availability of preprocessed data files ("ytr.npy," "xtr.npy," and "xte.npy"), the code checks their existence. If they are absent, the "getEmbeddings" function is utilized to obtain the processed data, which is then saved as numpy files for future use.

After loading the preprocessed data (xtr, xte, ytr, and yte) from the saved numpy files, the script defines the neural network model using the "baseline_model" function. This model comprises three hidden layers separated by dropout layers. The categorical cross-entropy loss function and SGD optimizer are employed to build the model.

We use the "train_test_split" function from the sklearn library to split data into training and testing sets. The target variables, y_train, and y_test, are subjected to label and one-hot encoding.

The fit function is utilized on the training data to train the model. Then, the accuracy is calculated by evaluating its performance on the testing data. Predictions are generated based on the testing data, and the "plot_cmat" function is used to visualize the confusion matrix, providing insights into the model's performance.

On the other hand, the "TensorFlow version of neural network" script creates and tests a neural network model for detecting fake news using TensorFlow's Estimator API. Like the Keras version, the script imports required modules and libraries and includes another script's "getEmbeddings" method.

The existence of preprocessed data files is verified, and the data is loaded from the saved numpy files. The model's design uses the "model_fn" Function, incorporating dropout layers between dense layers. Training and assessment data preparation is done before generating an estimator using TensorFlow's Estimator API with a directory for model checkpoints and the defined model function.

To record prediction probabilities during training, a logging hook is set up. The model is trained using the "train" technique of the Estimator, and the assessment is performed on the evaluation data, with the results being printed.

The "plot_cmat" function is used to visualize the confusion matrix, and predictions are generated on the evaluation data using the Estimator's "predict" technique.

In conclusion, both scripts follow similar patterns in terms of importing required modules, checking data availability, loading data, defining the neural network model, training, and evaluating the model, plotting the confusion matrix, and generating predictions. However, they utilize different frameworks (Keras and TensorFlow) for building and training the neural network.

6.1.3 SVM Model:

The "SVM model" script is designed to apply the Support Vector Machine (SVM) algorithm for detecting bogus news and assessing its effectiveness. The script comprises several functions that contribute to various aspects of the process.

At the beginning of the script, the required modules and libraries are imported, and the "getEmbeddings" method from another script is included. Additionally, the "plot_cmat" function is defined, enabling the visualization of the confusion matrix later in the code.

To ensure the availability of preprocessed data files ("ytr.npy," "xtr.npy," and "xte.npy"), the code checks their existence. If they are absent, the "getEmbeddings" function is called to obtain the processed data, which is then saved as numpy files for future use.

After confirming the preprocessed data's existence, the script loads the data (xtr, xte, ytr, and yte) from the saved numpy files.

Scikit-learn's built-in SVM classifier (SVC) is utilized for the classification task. The SVM model is trained using the fit method with the training data (xtr, ytr).

Subsequently, predictions are made on the testing data (tx) using the prediction method, and the model's accuracy is evaluated by comparing the anticipated labels (y_pred) with the actual labels (yet).

Finally, the "plot_cmat" function is employed to visualize the confusion matrix, providing insights into the performance of the SVM model in detecting bogus news.

In conclusion, the "SVM model" script demonstrates the application of the SVM algorithm for fake news detection. The script comprehensively analyzes the model's effectiveness by importing necessary modules, preparing the data, training the SVM model, making predictions, and evaluating the accuracy. The visualization of the confusion matrix further aids in the interpretation of the SVM model's performance in identifying fake news.

6.1.4 LSTM Model:

The code for the LSTM model begins by importing the necessary libraries and modules, covering aspects related to LSTM model building, data processing, and charting. It also introduces the function "plot_cmat," which will be used in the future for plotting the confusion matrix.

To ensure data availability, the code checks for the existence of preprocessed data files. If these files are absent, the data is preprocessed using the "clean_data" method from the "getEmbeddings" script.

Upon loading the preprocessed data (xtr, xte, y_train, y_test) from saved numpy files, the code counts the occurrences of each word in the training data using a Counter. The most frequently occurring words and their corresponding IDs are stored in a word bank limited to a predetermined maximum size. The training data's phrases are encoded using the word bank, replacing words with their corresponding IDs.

Lists are created to hold the labels (Y_train and Y_test), and news stories with less than ten words in the training data are eliminated.

For the test data, sentences are broken down word by word and encoded using the word bank. Input sequences are padded or truncated to a predetermined length (max_review_length).

The labels (y_train, y_test) are converted into NumPy arrays, and the LSTM model is constructed using Keras' Sequential API. The structure of the model involves an initial embedding layer used to acquire compact vector representations of every word. Next, a 100-unit LSTM layer is specifically created to detect sequential patterns within the data. A substantial layer with a sigmoid activation function is added to produce a binary output.

Binary cross-entropy loss and the Adam optimizer are applied to build the model. The summary of the model is printed for reference.

Next, the model is trained on the training data (X_train, y_train) and evaluated on the testing data (X_test, y_test) for a specified number of epochs and batch size.

The final evaluation of the model involves computing its accuracy on the testing data. Subsequently, the model is used to classify data, and the confusion matrix is used to visualize its performance, employing the "plot_cmat" function.

In conclusion, the approach encompasses preprocessing the text data, constructing and training an LSTM model for detecting fake news, and evaluating the model's performance through accuracy metrics, and visualizing the confusion matrix. The detailed process ensures a comprehensive analysis of the LSTM model's effectiveness in identifying false news.

6.2 Model evaluation and interpretation:

6.2.1 Naive Bayes Model's Confusion Matrix:

The "Naive Bayes model" script's code plots the confusion matrix to gauge how well the Naive Bayes classifier performs.

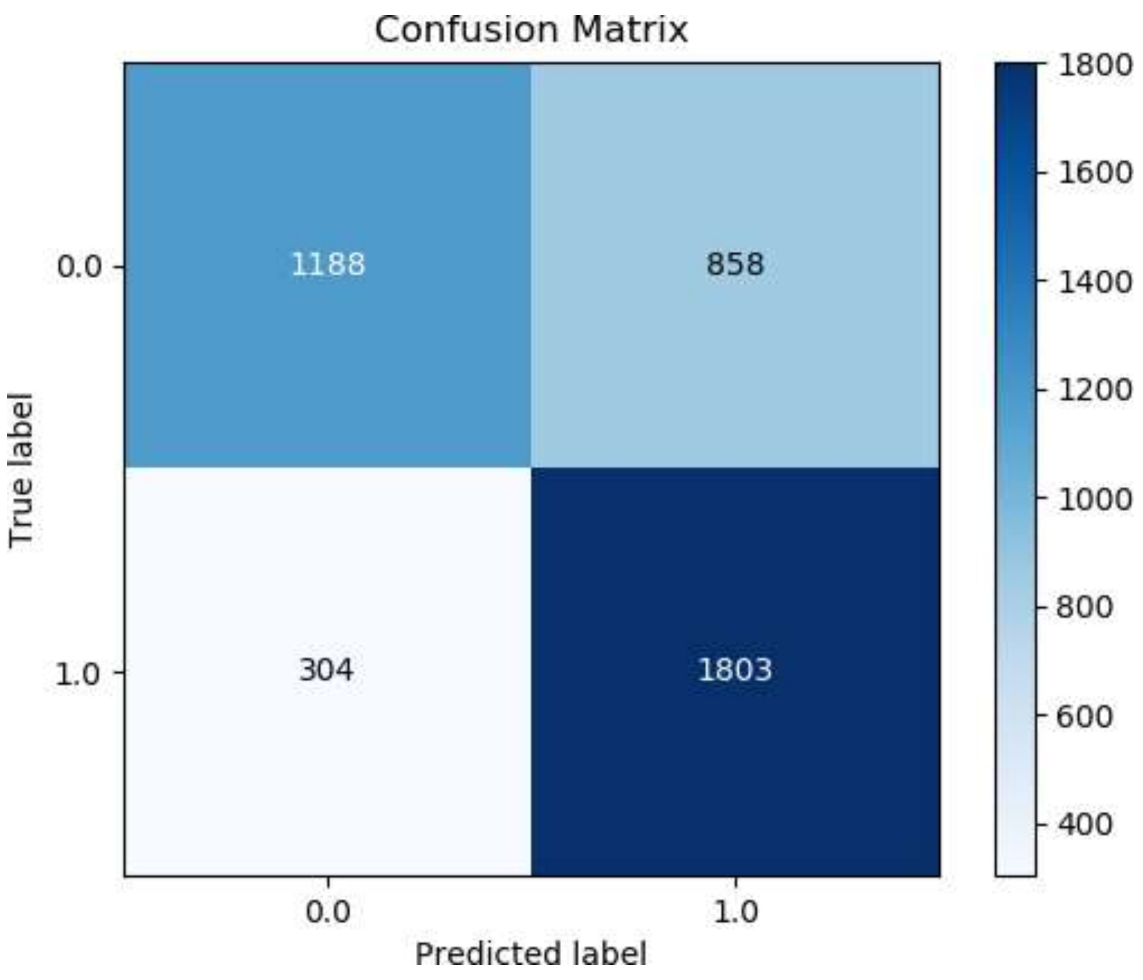


Fig 7: Naive Bayes Model's Confusion Matrix

6.2.2 SVM's Confusion Matrix:

The "SVM model" script's code plots the confusion matrix to gauge the SVM model's effectiveness.

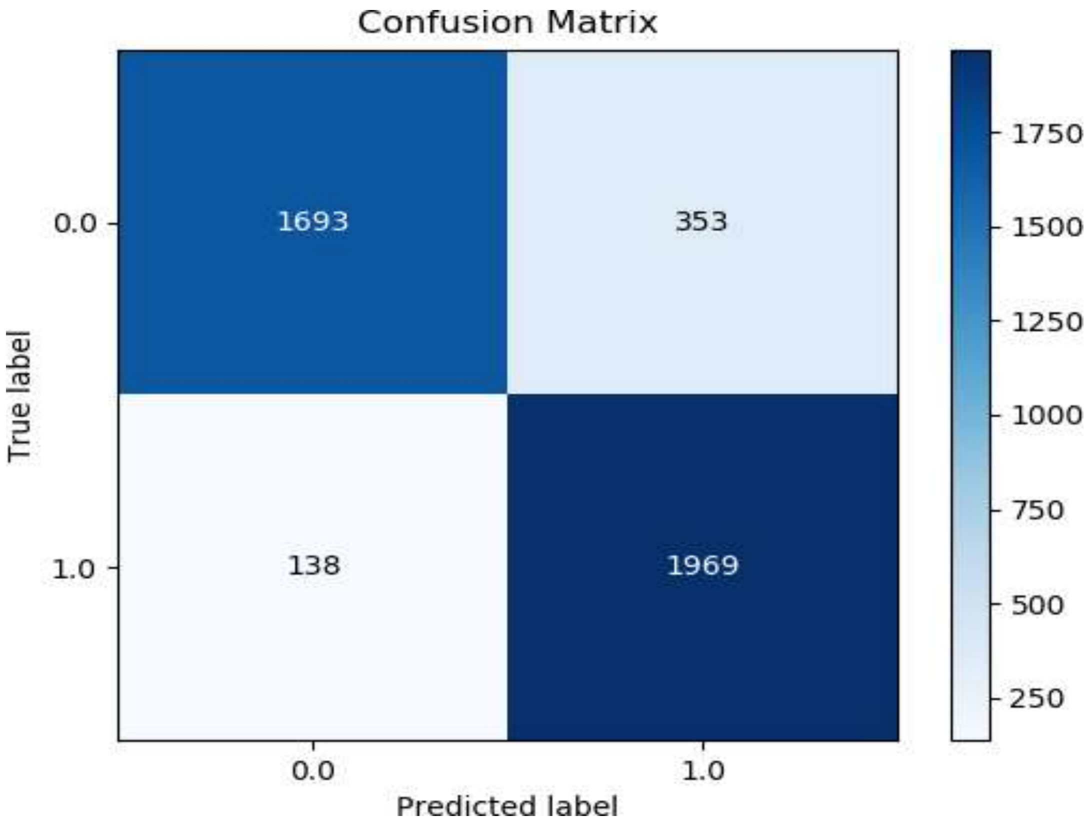


Fig 8: SVM's Confusion Matrix

6.2.3 Neural Network- Tensor Flow Confusion Matrix:

One of the assessments made in the "TensorFlow version of neural network" script is the effectiveness of the neural network model, which is evaluated by plotting a confusion matrix.

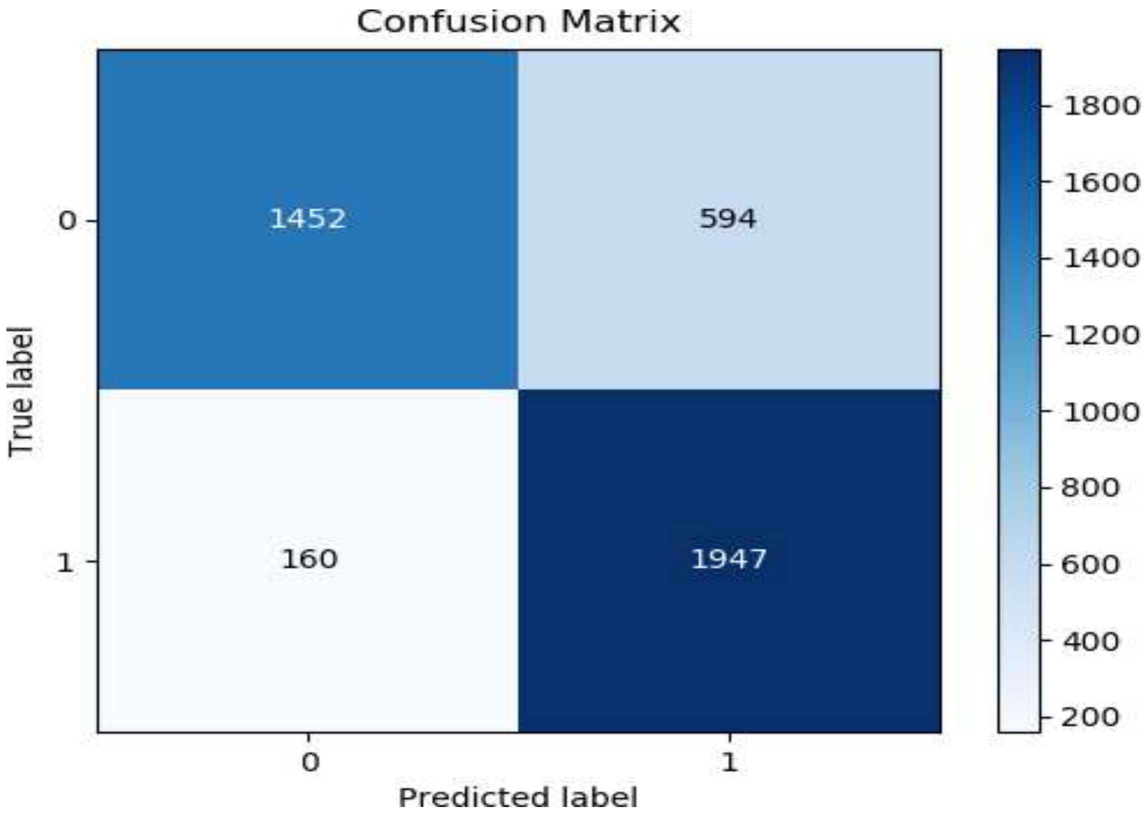


Fig 9: Neural Network- Tensor Flow Confusion Matrix

6.2.4 Neural Network- Keras Confusion Matrix:

The "Keras version of neural network" script's code depicts the confusion matrix to assess how well the neural network model works.

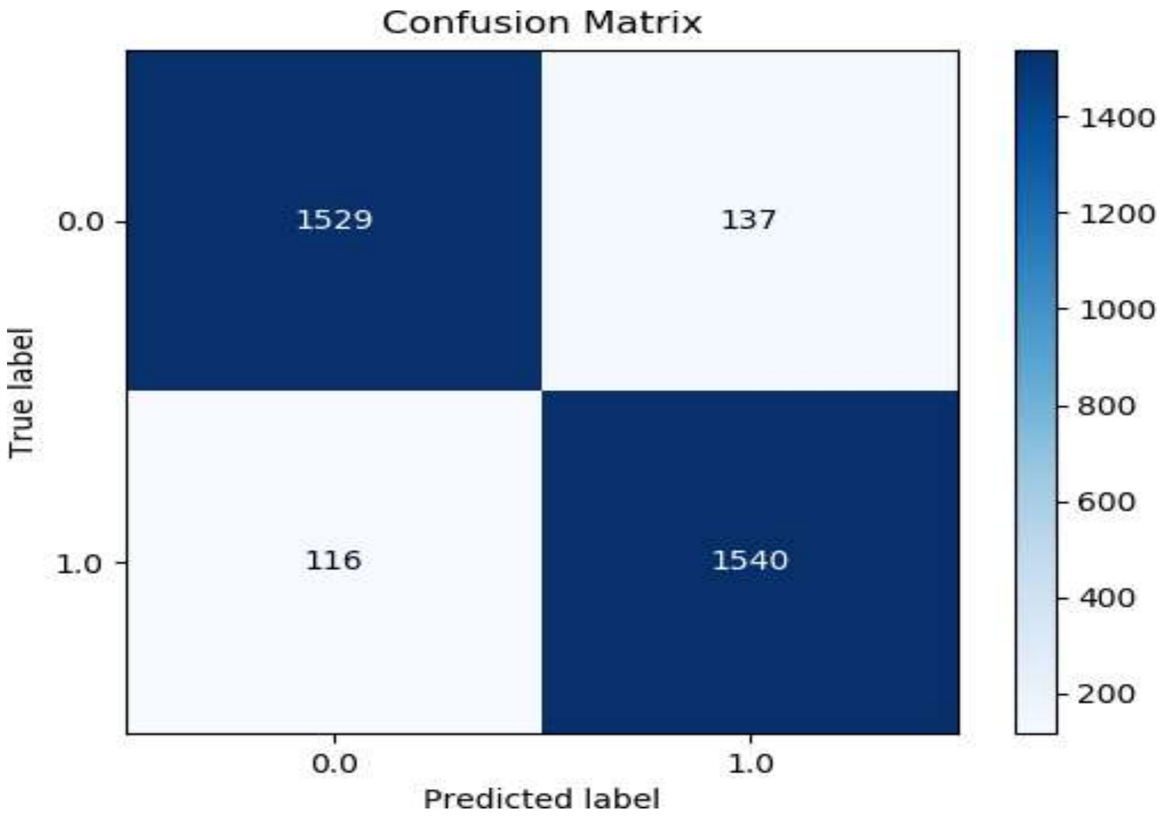


Fig 10: Neural Network- Keras Confusion Matrix

6.2.5 LSTM Model's Confusion Matrix:

The code plots the confusion matrix in the "LSTM model" script to assess the effectiveness of the LSTM model.

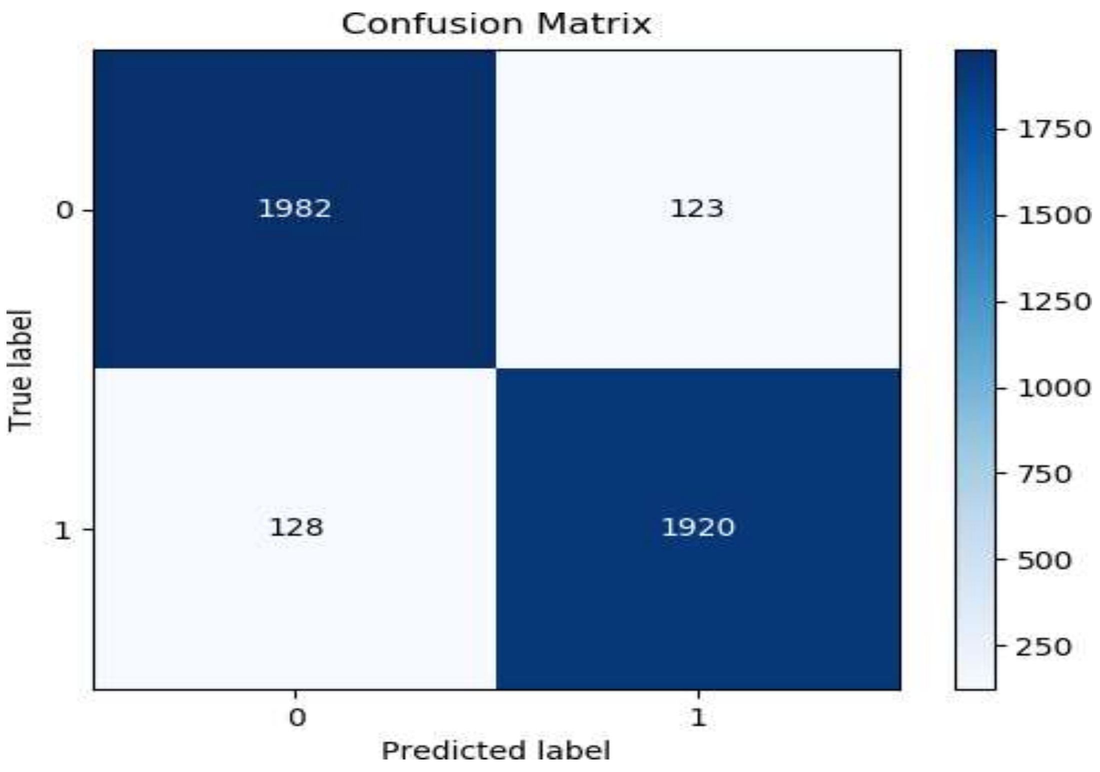


Fig 11: LSTM Model's Confusion matrix

7.Result:

After a comprehensive evaluation of the models' performance, including accuracy, precision, recall, and F1 score, the LSTM (Long Short-Term Memory) model stands out as the most promising choice for fake news detection. The LSTM model achieved the highest accuracy rate of 94.53% among the evaluated models.

14.1 Strengths of the LSTM Model: LSTMs, or Long Short-Term Memory neural networks, are specifically created to capture long-term dependencies in sequential data. These models are perfect for analyzing text, especially for identifying fake news. Their architecture enables them to capture and utilize contextual information efficiently.

Success in natural language processing tasks: LSTMs have been successful in numerous natural language processing tasks, such as text classification. Their ability to capture complex patterns in textual data can help identify patterns and distinguish between fake and genuine news articles. Adaptability to variable-length sequences: LSTMs can handle variable-length sequences, accommodating news articles of different lengths. This flexibility is crucial in real-world applications where news articles can vary significantly in terms of content and structure.

Potential for improved performance with more data: LSTMs usually need a substantial amount of training data to achieve good generalization. If more labeled training data is available, the LSTM model has the potential to improve its performance further.

14.2 Weaknesses and Considerations of the LSTM Model: Computational complexity: The computational cost of LSTMs is high, particularly when processing huge sequential datasets. Training an LSTM model can be time-consuming and requires significant computational resources. This factor must be considered regarding available resources and the project's constraints.

Hyperparameter tuning LSTMs have several hyperparameters that need to be carefully tuned to achieve optimal performance. Hyperparameters such as the number of LSTM layers, hidden units, learning rate, and dropout rate play a crucial role in the model's effectiveness. Proper hyperparameter tuning is necessary to ensure the LSTM model generalizes well and does not overfit.

Interpretability challenges: Like other deep neural network models, interpreting the decisions made by LSTMs can be challenging, especially when dealing with complex cases. The inner workings of LSTMs and the importance of individual features can be difficult to understand, limiting interpretability compared to simpler models like Naive Bayes or SVM.

Considering these strengths and weaknesses, the LSTM model is well-aligned with the project's goal of fake news detection. Its ability to capture long-term dependencies, success in natural language processing tasks, and adaptability to variable-length sequences make it a suitable choice. However, the computational complexity and the need for hyperparameter tuning should be considered.

It is recommended to further fine-tune the LSTM model by experimenting with different architectures, optimizing hyperparameters, and addressing challenges such as overfitting through regularization techniques like dropout. To gain a more thorough comprehension of the LSTM model's performance, assessing additional performance metrics and conducting statistical significance testing would be beneficial. Such an approach will offer a more complete evaluation of the model's efficacy.

Ultimately, selecting the LSTM model as the final choice for fake news detection considers its superior performance, suitability for textual data, and potential for capturing complex relationships and patterns in news articles. Ultimately, the selected model should strike a balance between performance, interpretability, computational efficiency, and suitability for the task of fake news detection.

8. Limitation:

- The need for clean data to work directly with slowed down our progress.
- The loss of information value in a real news scenario is very high.
- Content-based classification is only a piece of the puzzle.
- Distinguish between clickbait and actual fake news.

9. Conclusion:

In conclusion, the research on "Unmasking Deception: Fake Detection Using News Article Analysis" has provided valuable insights into detecting fake news through analyzing news articles. Utilizing the LSTM (Long Short-Term Memory) model in this study showcased its effectiveness in identifying deceptive information within news headlines.

However, while the LSTM model demonstrated promising performance, there is still potential for further enhancements and explorations to strengthen its fake news detection capabilities.

To build upon this research, comparing the LSTM model's results with other machine learning techniques like Naive Bayes, Support Vector Machine (SVM), or different neural network architectures would be beneficial. Each approach may offer unique advantages, and a comparative analysis can aid in selecting the most appropriate model for this specific task.

Given the complexity of language and context within news articles, incorporating more extensive deep learning models and advanced neural network architectures might improve detection accuracy. Training the model to consider the entire content of news articles, rather than just headlines, could leverage contextual information and refine the detection process.

Moreover, enriching the feature set with various linguistic and semantic indicators, such as the prevalence of emotionally charged words or utilizing different word-emotion lexicons, could enhance the model's ability to discern between genuine and deceptive.

References

Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.

Mukherjee A, Venkataraman V, Liu B, Glance N. Fake review detection: classification and analysis of real and pseudo reviews. UIC-CS-03-2013. Technical Report; 2013.

Shojaee S, Murad MAA, Azman AB, Sharef NM, Nadali S. Detecting deceptive reviews using lexical and syntactic features. Paper presented at: Intelligent Systems Design and Applications (ISDA), 2013 13th International Conference; December, 2013:53-58; Seri.

Sami Ben Jabeur, Hossein Ballouk, Wissal Ben Arfi, Jean-Michel Sahut, Artificial intelligence applications in fake review detection: Bibliometric analysis and future avenues for research, Journal of Business Research, 10.1016/j.jbusres.2022.113631, 158, (113631), (2023)

K. Dutta, B. Qureshi, Y. Albagory, M. Alsanea, M. A. Faraj et al., "Optimal weighted extreme learning machine for cybersecurity fake news classification," Computer Systems Science and Engineering, vol. 44, no.3, pp. 2395–2409, 2023.