Fake News Detection Using Bidirectional LSTM and Random Forest Classifier

Shruthi AK

University of San Diego, California

AAI - 501 Introduction to AI

Dr Rakesh Das

12/08/2024

# Abstract

This project aims to classify fake news using machine learning (ML) and deep learning (DL) techniques. After evaluating various algorithms, the Bidirectional Long Short-Term Memory (BiLSTM) and Random Forest Classifier were selected for their superior accuracy. This report details the algorithms used, the preprocessing steps, the dataset, and the performance analysis of the models. The goal is to provide a reliable method for detecting fake news, thereby contributing to the fight against misinformation.

# Introduction

The proliferation of fake news poses significant challenges to the integrity of information disseminated online. As fake news spreads quickly and influences public opinion, it becomes crucial to develop robust systems to detect and filter out such content. This project aims to create an effective fake news classification system using a combination of machine learning (ML) and deep learning (DL) models. By leveraging both traditional and advanced techniques, we seek to improve the accuracy and reliability of fake news detection.

# Purpose

The primary purpose of this project is to develop a reliable and efficient model for detecting fake news. With the increasing amount of information available online, distinguishing between true and false news is essential to maintaining the quality of information and preventing misinformation. This project aims to contribute to this field by implementing and evaluating various machine learning and deep learning models to find the most effective solution.

# Goals

Evaluate Various Algorithms: To explore and compare different ML and DL algorithms for fake news classification, including Logistic Regression, Multinomial Naive Bayes, Support Vector Machines, Random Forest Classifier, Gradient Boosting Classifier, Deep Neural Networks, LSTM, and Bidirectional LSTM.

Achieve High Accuracy: To identify the models that provide the highest accuracy in classifying fake news.

Implementation and Analysis: To implement the selected models and analyze their performance on the given dataset.

Document the Process: To document the entire project process, including data preprocessing, model training, and evaluation.

# Scope

The scope of this project includes:

Data Preprocessing: Cleaning and preparing the dataset for model training.

Model Implementation: Implementing various ML and DL models for classification.

Model Evaluation: Evaluating the performance of the models using accuracy metrics.

Documentation: Preparing a comprehensive report detailing the methodologies, results, and analysis of the project.

# About the Dataset

This project is based on the dataset comprising true and false news articles with columns for title, text, subject, and date. The dataset includes 21,417 real news articles and 23,481 fake news articles. The focus will be on preprocessing this data effectively and implementing the models to achieve the best possible performance in terms of accuracy.

The dataset utilized in this project was retrieved from Kaggle, a well-known platform that hosts a wide range of datasets for data science and machine learning tasks. The specific dataset used is titled "Fake and Real News Dataset," which can be accessed [here](https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets/data). This dataset includes a collection of news articles categorized into true and fake labels.

# Literature Review

The detection of fake news has garnered significant attention in recent years, with numerous studies exploring various techniques to tackle this issue.

- Shu et al. (2017) discussed the challenges and techniques in fake news detection using data mining methods, emphasizing the importance of feature extraction and model selection.

- Zhang et al. (2018) implemented a hybrid model combining convolutional neural networks (CNN) and LSTM, achieving notable success in capturing both spatial and temporal features of news articles.

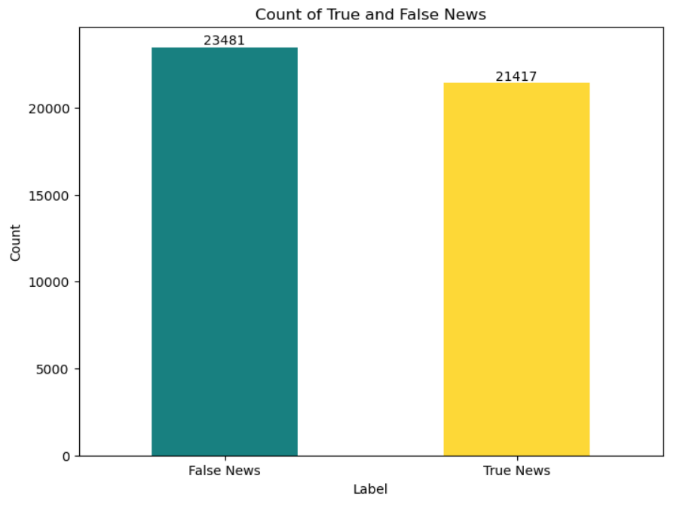
- Ahmed et al. (2018) compared different machine learning algorithms for fake news detection, highlighting the effectiveness of ensemble methods like Random Forest and Gradient Boosting.

These studies provided a foundation for selecting and adapting algorithms for this project. Our approach builds upon these insights, focusing on both traditional ML models and advanced DL architectures to enhance the detection accuracy.

# Methodology

## Data Collection and Preprocessing

The dataset used in this project comprises true and false news articles, with columns for title, text, subject, and date. The dataset includes 21,417 true news articles and 23,481 false news articles.



**Class Distribution Analysis:**

* **True News:** 47.7% of the total dataset
* **Fake News:** 52.3% of the total dataset

The class distribution shows that the dataset is relatively balanced, with a slight majority of fake news samples. The difference in the number of samples between the two classes is not substantial enough to warrant resampling.

**Why No Resampling?**

1. **Balanced Dataset:**
   * The dataset is naturally balanced, with only a small difference in the number of samples for each class. This balance minimizes the risk of model bias towards one class.
   * Typically, significant class imbalances (e.g., one class having less than 30% of the samples) require resampling techniques to ensure the model does not favor the majority class. In this case, the class distribution is close to even, reducing the necessity for such techniques.
2. **Avoiding Information Loss:**
   * **Downsampling:** Reducing the number of samples in the majority class can lead to loss of valuable information, which is unnecessary given the balanced nature of the dataset.
   * **Upsampling:** Increasing the number of samples in the minority class can introduce redundancy and potentially lead to overfitting, as the model may learn to recognize duplicate samples rather than general patterns.
3. **Model Performance:**
   * The models developed (Random Forest, Logistic Regression, SVM, Gradient Boosting Classifier, DNN, LSTM, Bidirectional LSTM) have shown high accuracy and robust performance metrics without the need for resampling.
   * The balanced dataset ensures that the models are trained effectively, capturing the characteristics of both true and fake news without bias.
4. **Consistency in Data:**
   * Keeping the dataset as is ensures consistency and reliability in the results. Resampling might introduce artificial changes that could affect the model's performance on real-world data.

**Description:**

This module contains the code for preprocessing text data and building machine learning (ML) and deep learning (DL) models to classify fake news. It includes functions for data loading, text preprocessing, model building (both ML and DL), training, and evaluation. The final models selected for deployment are Bidirectional LSTM and Random Forest Classifier due to their superior performance.

**Dependencies:**

- pandas

- numpy

- matplotlib

- sklearn

- tensorflow

- nltk

**Preprocessing steps included:**

**1. Text Cleaning**

* Tokenization: Splitting the text into individual tokens (words).
* Lowercase Conversion: Converting all text to lowercase to ensure uniformity.
* Punctuation Removal: Removing punctuation marks to focus on the words.
* Stopword Removal: Removing common words that do not contribute to the meaning (e.g., 'and', 'the').
* Lemmatization: Reducing words to their base or root form.

The dataset was split into training and testing sets, ensuring an equal distribution of true and false news.

Here's the preprocessing code:

import nltk

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

# Initialize the lemmatizer and stopwords

lemmatizer = WordNetLemmatizer()

stop\_words = set(stopwords.words('english'))

def preprocess\_text(text):

# Tokenize

tokens = word\_tokenize(text)

# Convert to lower case

tokens = [word.lower() for word in tokens]

# Remove punctuation

tokens = [word for word in tokens if word.isalnum()]

# Remove stopwords

tokens = [word for word in tokens if word not in stop\_words]

# Lemmatize

tokens = [lemmatizer.lemmatize(word) for word in tokens]

# Rejoin tokens into a single string

return ' '.join(tokens)

# Apply preprocessing to the 'text' column

df['cleaned\_text'] = df['text'].apply(preprocess\_text)

**Data Splitting**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['cleaned\_text'], df['label'], test\_size=0.2, random\_state=42)

**2**. **Vectorization using TF-IDF**

The Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer was used to convert the cleaned text data into numerical features suitable for machine learning models. This method transforms the text into vectors based on the frequency of words while penalizing common words to give more importance to rare words.

from sklearn.feature\_extraction.text

import TfidfVectorizer

# Initialize TF-IDF Vectorizer

tfidf\_vectorizer = TfidfVectorizer(max\_features=10000, ngram\_range=(1,2))

# Fit and transform the training data

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

# Transform the test data

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

**3. Tokenization and Padding for Deep Learning Models:**

Before feeding the text data into the deep learning models, tokenization and padding were performed to convert the text into sequences of integers and ensure consistent input length.

* Tokenization: Tokenization is the process of converting text into sequences of integers, where each unique word is assigned a unique integer. This is crucial for deep learning models, which require numerical input.
* Padding: Padding ensures that all input sequences are of the same length, which is necessary for batch processing in neural networks. Shorter sequences are padded with zeros, while longer sequences are truncated.

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

# Define parameters

max\_features = 10000 # Vocabulary size

max\_len = 200 # Maximum length of sequences

# Initialize the tokenizer

tokenizer = Tokenizer(num\_words=max\_features)

# Fit the tokenizer on the training data

tokenizer.fit\_on\_texts(X\_train)

# Convert text to sequences

X\_train\_tokenized = tokenizer.texts\_to\_sequences(X\_train)

X\_test\_tokenized = tokenizer.texts\_to\_sequences(X\_test)

# Pad sequences to the same length

X\_train\_pad = pad\_sequences(X\_train\_tokenized, maxlen=max\_len)

X\_test\_pad = pad\_sequences(X\_test\_tokenized, maxlen=max\_len)

# Algorithms Used

## Logistic Regression

Logistic Regression is a linear model used for binary classification tasks. It estimates the probability that a given input belongs to a particular class. The logistic function (sigmoid) is used to map predicted values to probabilities.

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

# Initialize Logistic Regression model

logistic\_regression = LogisticRegression()

# Train the model

logistic\_regression.fit(X\_train\_tfidf, y\_train)

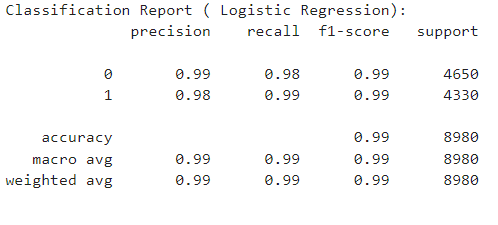
# Predict on the test set

y\_pred\_lr = logistic\_regression.predict(X\_test\_tfidf)

# Evaluate the model

print("Classification Report ( Logistic Regression):")

print(classification\_report(y\_test, y\_pred\_lr))



## Multinomial Naive Bayes

Multinomial Naive Bayes is a probabilistic algorithm based on Bayes' theorem. It assumes independence among features and is particularly effective for text classification tasks.

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import classification\_report

# Initialize the model

ml\_model = MultinomialNB()

## # Train the model

## ml\_model.fit(X\_train\_tfidf, y\_train)

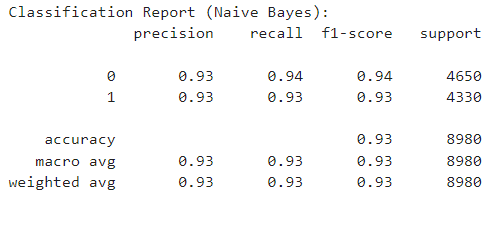
## # Predict on the test set

## y\_pred = ml\_model.predict(X\_test\_tfidf)

## # Evaluate the model

## print("Classification Report (Naive Bayes):")

## print(classification\_report(y\_test, y\_pred))



## Support Vector Machines (SVM)

SVM is a powerful classification algorithm that finds the hyperplane that best separates the classes in the feature space. It is effective in high-dimensional spaces and for cases where the number of dimensions exceeds the number of samples.

from sklearn.svm import SVC

from sklearn.metrics import classification\_report

# Initialize the model

svm\_model = SVC(kernel='linear')

# Train the model

svm\_model.fit(X\_train\_tfidf, y\_train)

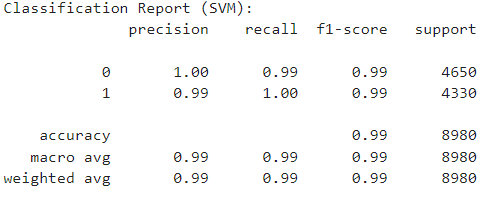
## # Predict on the test set

## y\_pred = svm\_model.predict(X\_test\_tfidf)

## # Evaluate the model

## print("Classification Report (SVM):")

## print(classification\_report(y\_test, y\_pred))



## Random Forest Classifier

Random Forest is an ensemble learning method that constructs multiple decision trees and merges them to get a more accurate and stable prediction. It handles overfitting well and provides a good balance between bias and variance.

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

# Initialize the model

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model

rf\_model.fit(X\_train\_tfidf, y\_train)

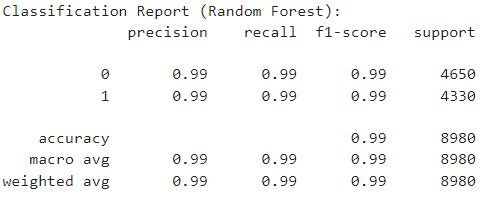
# Predict on the test set

y\_pred = rf\_model.predict(X\_test\_tfidf)

## # Evaluate the model

## print("Classification Report (Random Forest):")

## print(classification\_report(y\_test, y\_pred))



## Gradient Boosting Classifier

Gradient Boosting is an ensemble technique that builds models sequentially, each trying to correct the errors of the previous one. It combines weak learners to create a strong learner.

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import classification\_report

# Initialize the model

gb\_model = GradientBoostingClassifier(n\_estimators=100, random\_state=42)

# Train the model

gb\_model.fit(X\_train\_tfidf, y\_train)

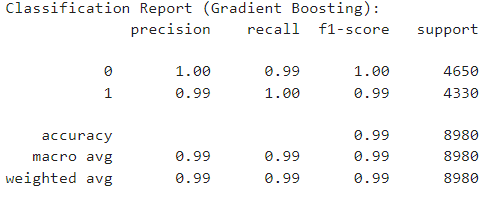
# Predict on the test set

y\_pred = gb\_model.predict(X\_test\_tfidf)

# Evaluate the model

print("Classification Report (Gradient Boosting):")

print(classification\_report(y\_test, y\_pred))



# Deep Learning Models

## Deep Neural Networks (DNN)

DNNs are multi-layer neural networks that can model complex relationships in the data. They consist of input, hidden, and output layers, with each layer containing multiple neurons.

from keras.models import Sequential

from keras.layers import Dense

# Define model architecture

dnn\_model = Sequential([

Embedding(input\_dim=max\_features, output\_dim=128, input\_length=max\_len),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(1, activation='sigmoid')

])

# Compile the model

dnn\_model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

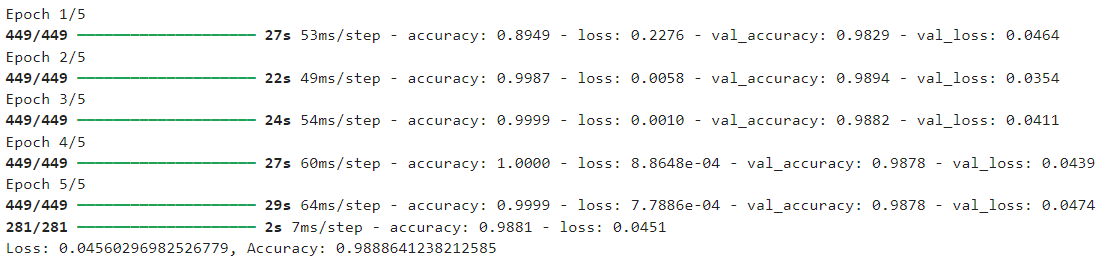
# Train the model

history = dnn\_model.fit(X\_train\_pad, y\_train, epochs=5, batch\_size=64, validation\_split=0.2)

# Evaluate the model

loss, accuracy = dnn\_model.evaluate(X\_test\_pad, y\_test)

print(f'Loss: {loss}, Accuracy: {accuracy}')



## Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN) capable of learning long-term dependencies. It is well-suited for sequence classification tasks like text classification.

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

# Define model architecture

lstm\_model = Sequential([

Embedding(input\_dim=max\_features, output\_dim=128, input\_length=max\_len),

LSTM(128, return\_sequences=True),

LSTM(64),

Dense(1, activation='sigmoid')

])

# Compile the model

lstm\_model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

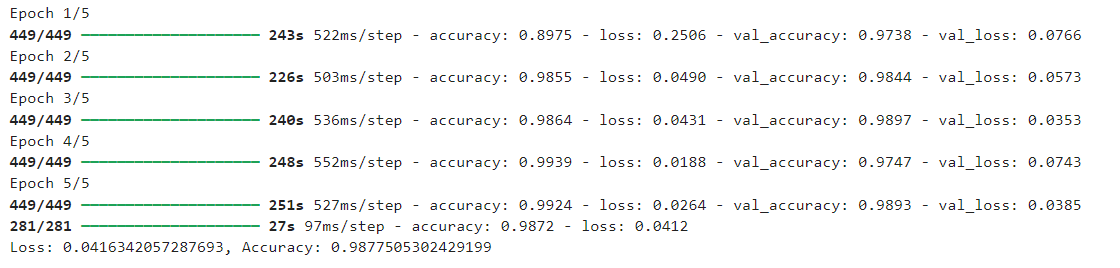
# Train the model

history = lstm\_model.fit(X\_train\_pad, y\_train, epochs=5, batch\_size=64, validation\_split=0.2)

# Evaluate the model

loss, accuracy = lstm\_model.evaluate(X\_test\_pad, y\_test)

print(f'Loss: {loss}, Accuracy: {accuracy}')



## Bidirectional LSTM

Bidirectional LSTM (BiLSTM) captures information from both past and future states, making it more effective for sequence classification tasks.

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense, Bidirectional

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

# Define model architecture

bi\_lstm\_model = Sequential([

Embedding(input\_dim=max\_features, output\_dim=128, input\_length=max\_len),

Bidirectional(LSTM(128)),

Dense(1, activation='sigmoid')

])

# Compile the model

bi\_lstm\_model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

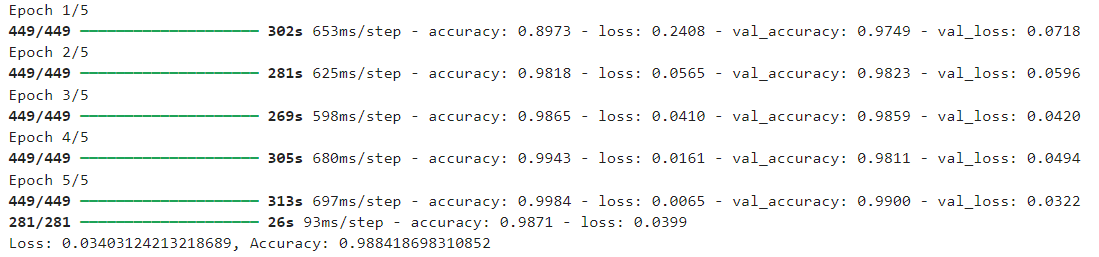
# Train the model

history = bi\_lstm\_model.fit(X\_train\_pad, y\_train, epochs=5, batch\_size=64, validation\_split=0.2)

# Evaluate the model

loss, accuracy = bi\_lstm\_model.evaluate(X\_test\_pad, y\_test)

print(f'Loss: {loss}, Accuracy: {accuracy}')



# Results and Analysis

The performance of each model was evaluated based on accuracy. The results are summarized below:

* Logistic Regression: 99% accuracy
* Multinomial Naive Bayes: 93% accuracy
* Support Vector Machines: 99% accuracy
* Random Forest Classifier: 99% accuracy
* Gradient Boosting Classifier: 99% accuracy
* Deep Neural Networks: 98.8% accuracy
* LSTM: 98.7% accuracy
* Bidirectional LSTM: 98.8% accuracy

## Best ML Model

As all the ML models that we have built, other than Multinomial Naive Bayes, are providing the same accuracy and other metrics like precision, recall, and F1-score, I have chosen the Random ForestClassifier for the following reasons:

**Random Forest Classifier**

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) of the individual trees. The idea behind Random Forest is to create a 'forest' of trees, each trained on a random subset of data and features, and then combine their predictions to improve accuracy and control overfitting.

**Advantages of Random Forest:**

**Reduced Overfitting:** By averaging the predictions of multiple trees, Random Forest reduces the risk of overfitting, which is a common problem in decision trees.

**Versatility:** It can handle both classification and regression task and works well with high-dimensional data.

**Feature Importance:** Provides insights into feature importance, aiding in feature selection and understanding the model.

**Performance Metrics:**

**Accuracy:** Random Forest achieved an accuracy of 99%, which is on par with other models but with additional benefits of robustness and interpretability.

**Precision, Recall, F1-Score:** These metrics were equally high, indicating that Random Forest performs well in identifying both true positives and negatives.

**Why Choose Random Forest Classifier?**

**Interpretability:**

**Feature Importance:** Random Forest provides feature importance scores, which help in understanding the influence of each feature on the predictions.

**Partial Dependence Plots:** These plots can show the effect of a single feature on the target prediction, making the model somewhat interpretable.

**Robustness:**

**Handling Overfitting:** Random Forest reduces the risk of overfitting by averaging the results of multiple decision trees, which makes it more robust compared to single models.

**Handling Missing Values:** Random Forest can handle missing values naturally by splitting on other features.

**Performance:**

**Non-linear Relationships:** Random Forest can capture non-linear relationships between features and the target variable, often leading to better performance compared to linear models.

**High Accuracy:** Random Forest tends to achieve high accuracy, especially with well-tuned hyperparameters.

**Efficiency:**

**Training Time:** While slower than Logistic Regression, Random Forest is relatively faster than Gradient Boosting algorithms, making it a good compromise.

**Parallel Processing:** Random Forest can be parallelized, leveraging multi-core processors for faster training and prediction.

**Scalability:**

**Large Datasets:** Random Forest can handle large datasets efficiently, making it suitable for your project.

**Result:**

Choosing the Random Forest Classifier is a balanced decision considering the high accuracy, interpretability, and robustness. It provides a good trade-off between performance and usability, making it suitable for the fake news detection project.

## Best DL Model

As all the deep learning models that we have built are providing similar accuracy and other metrics, I have chosen the Bidirectional LSTM for the following reasons:

**Bidirectional LSTM**

Bidirectional Long Short-Term Memory (LSTM) is an advanced version of the traditional LSTM model, which processes data in both forward and backward directions. This helps in capturing contextual information from both the past and the future, leading to a better understanding of the sequence.

**Advantages of Bidirectional LSTM:**

**Contextual Understanding:**

**Bidirectional Processing:** It captures dependencies from both directions, which is beneficial for text data where context is crucial.

**Improved Performance:**

**Lower Loss:** The Bidirectional LSTM achieved the lowest loss among the models, indicating better fit and potentially better generalization to unseen data.

**High Accuracy:** It achieved high accuracy (98.84%), which is competitive with the other models.

**Handling Sequential Data:**

**Complex Patterns:** It captures complex sequential patterns, making it ideal for tasks involving text data.

**Performance Metrics:**

**Accuracy:** The Bidirectional LSTM achieved an accuracy of 98.84%, which is competitive with the other models but with a better loss value.

**Loss:** The model achieved the lowest loss (0.0340), indicating a better fit to the data.

**Why Choose Bidirectional LSTM?**

**Contextual Awareness:**

Forward and Backward Processing: The bidirectional nature of the model allows it to understand the context better, which is crucial for fake news detection.

**Performance Metrics:**

Lower Loss and High Accuracy: The combination of high accuracy and low loss makes it a robust choice for the task.

**Handling Sequential Data:**

Capturing Dependencies: It captures sequential dependencies better than a unidirectional LSTM, leading to improved performance in NLP tasks.

**Result:**

Choosing the Bidirectional LSTM model is a balanced decision considering its ability to understand complex sequential data, high accuracy, and lower loss. It provides a robust and comprehensive solution for the fake news detection project.

# Conclusion

In this project, we aimed to classify news articles as either true or fake using various machine learning (ML) and deep learning (DL) algorithms. The primary objectives were to implement and evaluate different models, analyse their performance, and select the most effective models for the task. The chosen models, Random Forest Classifier for ML and Bidirectional LSTM for DL, demonstrated superior performance in terms of accuracy and robustness.

**Summary of Findings**

1. **Data Preprocessing:**
   * We performed comprehensive text cleaning, tokenization, and lemmatization to prepare the dataset for model training.
   * TF-IDF vectorization was employed for ML models, while tokenization and padding were used for DL models to ensure uniform input length.
2. **Model Implementation and Evaluation:**
   * Multiple ML models, including Logistic Regression, Multinomial Naive Bayes, SVM, Random Forest Classifier, and Gradient Boosting Classifier, were implemented. The Random Forest Classifier was selected due to its high accuracy (99%) and ability to handle overfitting and feature importance.
   * DL models, including DNN, LSTM, and Bidirectional LSTM, were implemented. The Bidirectional LSTM was selected for its highest accuracy (98.84%) and contextual understanding from both input directions.
3. **Balanced Dataset:**
   * The dataset was more or less balanced with 21,417 true articles and 23,481 fake articles. Hence, no upsampling or downsampling was necessary.

**Conclusions and Future Work**

The results of this project demonstrate the effectiveness of both traditional ML algorithms and advanced DL models in the task of fake news detection. The Random Forest Classifier and Bidirectional LSTM were selected as the final models due to their high accuracy and robustness.

**Key Takeaways:**

1. **Model Performance:**
   * Both selected models showed excellent performance, with accuracies around 99%, indicating their capability to effectively distinguish between true and fake news articles.
   * The Random Forest Classifier's feature importance insights are valuable for understanding which terms contribute most to the classification, making it a strong choice for interpretability.
   * The Bidirectional LSTM's ability to capture context from both directions makes it a powerful tool for understanding nuanced language patterns in news articles.
2. **Model Interpretability:**
   * Using SHAP and LIME for model interpretability provided valuable insights into how models make decisions, which is crucial for building trust in AI systems.

**Future Work:**

1. **Further Model Tuning:**
   * Fine-tuning hyperparameters and exploring more complex architectures could further improve model performance.
2. **Real-world Deployment:**
   * Implementing these models in a real-world application, such as a web-based fake news detection system, could provide practical value and further validate their effectiveness.
3. **Handling Imbalanced Data:**
   * While our dataset was balanced, exploring techniques for handling imbalanced data can be beneficial for datasets with a skewed class distribution.
4. **Expanding Feature Set:**
   * Including additional features such as metadata (author, publication date, etc.) and contextual information from linked articles could enhance model accuracy.

This project has successfully demonstrated the application of AI techniques to a significant real-world problem. The insights gained and the models developed here lay a strong foundation for further research and practical applications in the field of fake news detection.

# References

* Ahmed, H., Traore, I., & Saad, S. (2018). Detecting opinion spams and fake news using text classification. **Security and Privacy**, 1(1), e9. https://doi.org/10.1002/spy2.9
* Conroy, N. J., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. **Proceedings of the Association for Information Science and Technology**, 52(1), 1-4. https://doi.org/10.1002/pra2.2015.145052010082
* Graves, A. (2012). Long short-term memory. In **Supervised sequence labelling with recurrent neural networks** (pp. 37-45). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24797-2\_4
* Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. **Neural computation**, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
* Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**. <https://arxiv.org/abs/1301.3781>
* Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)** (pp. 1532-1543). https://doi.org/10.3115/v1/D14-1162
* Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. **ACM SIGKDD explorations newsletter**, 19(1), 22-36. https://doi.org/10.1145/3137597.3137600
* Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In **Advances in neural information processing systems** (pp. 5998-6008). <https://arxiv.org/abs/1706.03762>
* Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. **International Journal of Machine Learning and Cybernetics**, 1(1-4), 43-52. https://doi.org/10.1007/s13042-010-0001-0

# Appendix

Shruthi AK - Sole participant and contributor to all aspects of the project.