Analysing Classifiers for Fake News Detection: Comparison between Traditional Methods and Recent Methods

Submitted by

Dongpin HU

A final project submitted in fulfillment of the requirements for

the Degree of Bachelor of Science

at The University of London

September 2023

Table of Contents

Chapter 1. Introduction

1.1 Background of the study

As news reading through social media has grown increasingly widespread, the issue of fake news has become a major concern for both the public and the government. False information can use multimedia content to deceive readers and spread rapidly, leading to negative consequences and potential manipulation of public events. The dissemination of false news through various multimedia channels can have a detrimental impact on the credibility of legitimate news sources. For example, fake news may distort election results in favour of wicked politicians or foreign interests.

Detecting fake news is a crucial task that serves the dual purpose of providing genuine information to users and preserving the integrity of the news ecosystem (Shu et al., 2019). Twenty years ago, it was impossible to manually distinguish between true and false information in a timely manner when dealing with a large amount of data (Tacchini et al., 2017). With the rise of artificial intelligence and machine learning, it is possible to address this problem through automated fake news detection (Wang et al., 2018). Although there are already some cutting-edge models like transformer for fake news detection, which heavily rely on computing power, it is too much expensive for personal users of AI models (Li et al., 2020). It is necessary to find an energy-efficient and effective classifier for personal users among commonly used classifiers and recent classifiers. Second, previous studies on this topic have predominantly relied on theoretical analysis of classifiers rather than practical implementation through code. It is necessary to implement different fake news classifiers through code and compare their performance under the same dataset and same running environment.

This project aims to fill the aforementioned two research gaps. In detail, this project will make comparisons between traditional methods (e.g., Naive Bayes, support vector machine, etc) and advanced methods developed in recent years including long short-term memory (LSTM), bidirectional LSTM, and convolutional neural network (CNN) for fake news detection in terms of accuracy, precision, recall, and f1-score.

This project uses the project template of ***CM3060 Natural Language Processing Project Idea Title 1: Fake news detection.***

1.2 Research gaps

Few studies have compared the performance of different machine learning approaches for fake news detection (Ahmed et al. 2021; Busioc et al., 2020; Manzoor & Singla, 2019; Zhou

& Zafarani, 2020). Ahmed et al.'s (2021) study only listed and described the common machine learning supervised classifiers for fake news detection like Naive Bayes, SVM, logistic regression, random forest, KNN, RNN, etc and does not compare each classifier in code implementation. Busioc et al.'s (2020) study reported the performance of different classifiers for fake news detection respectively, but these classifiers were implemented in different datasets and working environments, making it impossible to compare and find the best performance model for fake news detection. Similarly, Manzoor and Singla's (2019) study only described the feature of each classifier and does not indicate which classifier is more effective and accurate in detecting fake news in practice. Wang (2017) indicated that researchers should use a labeled benchmark data set named "LIAR" for the NLP task of fake news detection. By using the same dataset for all classifiers for fake news detection, the results can be easily compared because these classifiers are based on the same data. In addition, for reproducibility, researchers can reproduce others' work more easily if they use the same dataset, enabling the possibility of comparing results of different classifiers on fake news detection. In conclusion, these studies do not indicate which classifier for fake news detection performs the best under the same dataset and working environment.

Second, previous studies on this topic have predominantly relied on theoretical analysis of classifiers rather than practical implementation through code. The integration of theory and code implementation is crucial for finding the most effective solution to fake news detection. It is necessary to implement different fake news classifiers through code and compare their performance under the same dataset and same running environment.

1.3 Objective of the project

To make comparisons between traditional methods (i.e., Naive Bayes, support vector machine, logistic regression, random forest classifier, and k-nearest neighbors) and advanced methods developed in recent years including long short-term memory (LSTM), bidirectional LSTM, and CNN for fake news detection in terms of accuracy, precision, recall, and f1-score.

1.4 Significance of the study

Cutting-edge models like transformer for fake news detection usually require high computing power, making it much expensive for personal users and small companies to implement state-of-the-art AI models (Li et al., 2020). Smaller models usually can be trained in less time and may produce comparable performance to state-of-the-art large models (Tan & Le, 2021). Thus, it is meaningful to find energy-efficient and effective classifiers which have comparable model performance to advanced models.

1.5 Project concept

The scope of this project is to create a Jupyter Notebook file which can compare the performance of traditional methods and recent methods for fake news detection. As mentioned in Section 1.3, the objective of this project is to make comparisons between five traditional models and three advanced models for fake news detection in terms of accuracy, precision, recall, and f1-score. The expected deliverable is a Jupyter Notebook file, which can be used for comparison between these eight classifiers for fake news detection.

The timeline of this project is listed as a burndown chart. There are eight key dates. On May 1st, it was the project starting date when I chose the NLP template for fake news detection because I was always interested in detecting fake news in current age of fake news. On May 10th, I found the first dataset named ISOT and began using this dataset for the comparison of eight machine learning models for fake news detection. On May 20th, I successfully created five traditional models (e.g., Naive Bayes) on Jupyter Notebook. On June 1st, I compared the performance of these five models based on evaluation methods (i.e., accuracy and sklearn metrics) and wrote the midterm report including literature review and the evaluation results of these five models. On June 20th, that is the fifth important date when I found the second dataset named LIAR and I also created three advanced models. On July 10th, I evaluated these eight models based on the first and second datasets. On August 1st, I wrote the whole report for the final project. During the ten days before the end date on August 20th, I checked the details of this report and got feedback from tutors.

The estimated budget of this project is relatively low like 800 USD. This project only needs 100 hours of a programmer with four years' programming experience for coding and report writing (300 USD) and a desktop with common computing power (i.e., Dell Desktop F8II9HH and its price was 500 USD). Section 3.4 has included more information about the desktop.

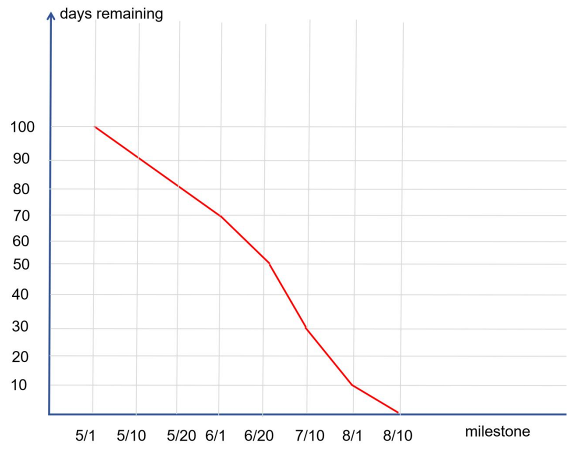

Figure 1. Burndown Chart of Final Project for Fake News Detection

Chapter 2. Literature Review

2.1 Fake news

Fake news refers to intentionally written content that is designed to mislead and deceive the general public (Mishra et al., 2022). Fake news detection is the process of differentiating between news articles that are true and those that are intentionally created to deceive readers (Pérez-Rosas et al., 2017). Fake news has three main characteristics: 1) the huge volume of fake news; 2) the variety of fake news; 3) the fast velocity of fake news (Zhang & Ghorbani, 2020).

Fake news detection usually involves implementing automated techniques such as machine learning and natural language processing to process news articles and extract features that reveal whether the article is genuine or fake (Zhou & Zafarani, 2020). In order to ensure that the public can access credible and reliable news, it has become very crucial for humans to develop robust fake news detection methods and thereby avoiding the negative consequences of fake news on society and humans.

The prevalence of fake news is pervasive, disturbing, and ubiquitous. It is causing extensive consequences for both individuals and society. As such, it is essential to develop a reliable and efficient fake news detection system to identify and thus combat its spread (Zhang & Ghorbani, 2020). Recently, detecting fake news has gained significant attention from scholars and companies, and numerous sociological studies have highlighted the impact of fake news and its influence on people's behaviors and reactions (Abu Arqoub et al., 2022; Andersen & Søe, 2020). Detecting fake news is a crucial task that serves the dual purpose of providing genuine information to users and preserving the integrity of the news ecosystem (Shu et al., 2019).

In the past, the approach of using expert-oriented fact-checking was proposed to distinguish between fake and true news. However, with the increasing amount of information on the Internet, it becomes increasingly difficult and even impossible for fake-news-checking professionals to differentiate fake and true news manually (De Beer & Matthee, 2021). This is where automated methods come in. In order to minimize the adverse effects of fake news, it is essential to create a mechanism that can identify it automatically as soon as it surfaces on social media (Mishra et al., 2022). By using machine learning techniques, automated fake news detection is possible to analyze a large amount of news and classify news into fake or true. Automated detection for fake news is preferable because automated detection for fake news is faster and more efficient than manual methods, and it can provide reliable results.

2.2 Traditional methods for fake news detection

There are five commonly used machine learning methods for fake news detection, namely Naive Bayes classifier, Support Vector Machine, Logistic Regression, Random Forest Classifier, and K-Nearest Neighbors (KNN).

Naive Bayes classifiers, used in machine learning, are a set of straightforward probabilistic classifiers that apply Bayes theorem while making naive assumptions of independence among the features. Naive Bayes classifiers do not constitute a single algorithm for training such models, but instead, a group of algorithms that share the same principle: all Naive Bayes classifiers assume that the value of a particular feature is independent of any other feature's value, given the class variable (Pattekari & Parveen, 2012). Naive Bayes classifiers have been widely employed for statistical techniques in email filtering, initially emerging in the mid-1990s as an effort to address the issue of spam filtering (Granik & Mesyura, 2017).

Support Vector Machine (SVM) can be defined as systems that employ a hypothesis space of linear functions in a feature space of high dimensionality, trained using an optimization-based learning algorithm that incorporates a learning bias derived from statistical learning theory (Jakkula, 2006). The principles of SVM were formulated by Vapnik (1999) and have become popular due to several attractive attributes, including improved empirical performance. Singh et al.'s (2017) study indicates that SVM performs the best prediction results among these five common classifiers for fake news detection.

Logistic Regression is a typical statistical method used to determine the probability of dependent variables (usually binary) based on the values of the independent variables (Domínguez-Almendros et al., 2011). The process is similar to that of multiple linear regression, except for the fact that the dependent variable is usually binary. Logistic regression is a commonly predictive method in various fields such as medical studies, advertising, and finance to predict outcomes.

Random Forest is a popular classification algorithm consisting of various decisions trees (Sharma et al., 2020). Random forest uses trees as base learners to produce a model that is both more reliable and precise (Cutler et al., 2012).

K-Nearest Neighbors (KNN) also known as nearest neighbor classification, is a type of classification algorithm that utilizes the concept of similarity between patterns (Kramer, 2013). The mechanism of KNN is to identify the K-nearest patterns to find a target pattern and thus determine the label. Then, KNN assigns the class label based on the K-nearest patterns in data space. To make this possible, it needs to define a similarity measure in data space to determine the closest patterns.

2.3 Long short-term memory (LSTM) for fake news detection

LSTMs are a type of recurrent neural network (RNN) that are particularly well-suited for processing sequential data, such as text (Bahad et al., 2019). LSTM usually acts as a timing layer in classification models due to its ability to store context history information and perform long-distance context-dependent learning (Bai, 2018).

LSTM is a gradient based method, which is novel and efficient in NLP tasks (Hochreiter & Schmidhuber,1997). LSTMs have a longer memory span than regular RNNs, which enables them to comprehend and exploit more intricate language dependencies (Sherstinsky, 2020). By discarding unnecessary data and noise in the input stream, LSTMs exhibit greater stability and adaptability to fluctuations and inconsistencies in language. LSTMs possess the ability to handle variable-length sequences, making them well-suited for classification tasks like sentiment analysis, text classification, and language translation (Staudemeyer & Morris, 2019). Studies have demonstrated that LSTMs perform better than conventional RNNs on various NLP applications, such as speech recognition, machine translation, and language modeling (Graves, 2012).

2.4 Bidirectional LSTM learning

A Bidirectional LSTM layer is a specific type of LSTM layer that can handle input sequences in both forward and backward directions (Graves et al., 2005). This layer consists of two parallel LSTM layers: one processes the input sequence from start to end, and the other processes it from end to start. The two output sequences from the parallel layers are then merged to create the final output.

Bidirectional LSTM layers are beneficial in modeling long-term dependencies in natural language processing because they can take into account both past and future contexts in language (Habernal & Gurevych, 2016). Consequently, they are also very commonly employed in various natural language processing tasks, such as sentiment analysis, text classification, and language translation.

2.5 Convolutional Neural Network (CNN)

CNNs consist of three types of layers: convolutional layers, pooling layers, and fully-connected layers (O'Shea & Nash, 2015). The function of neurons in CNNs is similar to those in traditional Artificial Neural Networks (ANNs). Each neuron receives an input and performs an operation, such as a scalar product followed by a non-linear function. However, CNNs are primarily used for pattern recognition in images, which allows image-specific features to be encoded into the architecture (Albawi et al., 2017). This makes CNNs more appropriate for image-focused tasks, while also reducing the number of parameters needed to set up the model.

2.6 Baseline performance

Based on literature in this field, the accuracy of each classifier mentioned in this study is listed as follows:

For Naive Bayes method, researchers achieved classification accuracy of 60% (Mhatre & Masurkar, 2021). For Support Vector Machine, the classification accuracy was 56% (Singh et al., 2017). For Logistic Regression, researchers achieved classification accuracy of 56% (Kaur et al., 2020). For Random Forest classifier, the classification accuracy was 65% (Sharma et al., 2020). For KNN method, researchers achieved classification accuracy of 54% (Mishra et al., 2022).

For LSTM, researchers achieved classification accuracy of 54% (Sansonetti et al., 2020). For Bidirectional LSTM, researchers achieved classification accuracy of 58% (Sansonetti et al., 2020). For CNN, researchers achieved classification accuracy of 58% (Samadi et al., 2020).

Performance of five common traditional machine learning methods for fake news detection is summarized in Table 1. In order to keep the table clean, I remove percent sign for each data.

Table 1. Performance of five traditional machine learning methods

model	accuracy score	precision	recall	f1-score
Naive Bayes	60	59	60	59
Support Vector Machine	56	56	56	48
Logistic Regression	56	56	56	51
Random Forest	65	69	83	75
KNN	54	54	54	54

Chapter 3. Design

3.1 Research design

This project uses a statistical technique called controlling for a variable. In order to keep conditions equally the same when running each model, I controlled several variables: running environment, running hardware, and dataset. The only difference is model selection. Thus, this project can compare each fake news classifier in a controlled condition.

The procedure of this comparative experiment is shown as below.

First, I loaded the first dataset named ISOT on Jupyter Notebook. Its code is:
df = pd.read_csv('C:/Users/Peter/Desktop/Final Project/Fake_True.csv', encoding='latin-1').

Second, I created the feature column and target column through code:
corpus = *df['text']*
y = df['whether_real']

Third, I split the data into training and testing sets using the train_test_split() function from scikit-learn. I used 80% of the data for training and 20% for testing. The code is: *X_train, X_test, y_train, y_test = train_test_split(corpus, y, test_size=0.2, random_state=42)*

Fourth, I then created a TfidfVectorizer object named tfidf_vector and used it to transform the training and testing data into a matrix of token counts. I also removed stop words using the 'english' option. Its code is: *tfidf_vector = TfidfVectorizer(stop_words = 'english')*

X_train = tfidf_vector.fit_transform(X_train.values.astype('U'))

X_test = tfidf_vector.transform(X_test.values.astype('U'))

Fifth, I ran each the model of each classifier for fake news detection and generated accuracy report for each model. One example is shown as below: *nb_model = MultinomialNB()*

nb_model.fit(X_train, y_train)

y_nb_pred = nb_model.predict(X_test)

print("NB Accuracy Score:", accuracy_score(y_test, y_nb_pred))

Sixth, I evaluated each model using confusion matrix and sklearn metrics and generated performance reports in terms of precision, recall, and f1-score. Example code is shown as below: *nb_cm = confusion_matrix(y_test, y_nb_pred)* *plot_confusion_matrix(nb_cm, classes=['True News', 'Fake News'], title= "Naive Bayes Confusion Matrix")*

nb_classifier_report = classification_report(y_test, y_nb_pred)

print(nb_classifier_report)

Seventh, I repeated the first to sixth operation for the second dataset named LIAR.

Eighth, I made comparisons between each model in terms of four aspects based on the results from these two datasets respectively, namely accuracy, precision, recall, and f1-score.

3.2 Datasets

First, the dataset named ISOT Fake News Dataset is downloaded from the official website of University of Victoria and saved as a csv file for the required natural language processing task on the final project, namely fake news detection. The dataset comprises articles on various subjects, with the majority of them centering on political and global news topics. The genuine articles were obtained by crawling articles from Reuters.com, while the fake news articles were collected from flagged unreliable websites by Politifact and Wikipedia, respectively (University of Victoria, n.d.). The dataset has totally 44,829 rows with 23,414 true articles from reuters.com and 21,415 articles from different fake news outlet resources. Every article in the dataset includes the following details: title of the article, content of the article, article category and the date when the article was published. Although the collected data underwent cleaning and processing, the punctuation marks and errors that were present in the fake news articles were retained in the text.

Second, the dataset named LIAR Fake News Dataset is downloaded from the official website of paperswithcode.com. The LIAR dataset is a collection of fake news detection data that is accessible to the public (Wang, 2017). Over a period of ten years, 12,800 short statements were manually labeled and gathered from politifact.com. These statements cover a range of contexts and are accompanied by in-depth analysis reports and links to the source documents for each case. The dataset has totally 4,560 rows with 2,053 rows of true news and 2,507 rows of fake news.

3.3 Dependencies

First, it is necessary to install the required module before importing it. The code is shown as below:
!pip install tensorflow

Second, import all required modules, such as numpy, pandas, display, tensorflow, sklearn, keras, etc. The code is shown as below:

```
import numpy as np
import pandas as pd
from IPython import display
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, Embedding, Bidirectional, Conv1D,
GlobalMaxPooling1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

3.4 Running environment

This project is running as ipynb file on Jupyter Notebook, which is a web-based interactive computing platform. The computer is Dell Desktop F8II9HH. Its processor is Intel(R) Core(TM) i7-8700T CPU @ 2.40GHz, and its RAM is 32 GB. This computer can be regarded as a common computer for personal users because it does not have high processing performance and it does not have a discrete graphics card.

3.5 Evaluation methods

3.5.1 Accuracy

Fake news detection is a type of classification task. When people think about evaluating a model's performance, the accuracy score is often the first metric that comes to mind. The formula for accuracy is shown as follows:

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Accuracy refers to the percentage of the correct prediction made by the classifier. According to the accuracy formula, accuracy is determined by dividing the number of correct predictions by the total number of predictions, as stated by Suresh (2020).

To comprehend the formula, it is necessary to have an understanding of TP, FP, TN, and FN. The succeeding section will clarify the definitions of these four acronyms.

True Positive (TP): both the actual value and prediction are positive.

True Negative (TN): both the actual value and prediction are negative.

False Positive (FP): when the actual value is negative but the prediction is positive, which is also referred to as a Type 1 error.

False Negative (FN): when the actual value is positive but the prediction is negative, which is also referred to as a Type 2 error.

It is commonly understood that relying solely on the accuracy score is insufficient to fully assess the performance of a model. Therefore, it is not recommended to solely use accuracy score as a means of evaluating model performance.Thus, adding a confusion matrix to the evaluation process can make the overall evaluation method more rational and empirical.

Confusion matrix is a useful tool in evaluating models as it presents a table of correct and incorrect predictions in a visual format. It is commonly used in machine learning, particularly for classification problems, because it covers all possible scenarios in a classification problem. The confusion matrix is typically 2 x 2, with the columns representing the predicted class and the rows representing the actual class. By presenting four dimensions of accuracy, it provides a more comprehensive evaluation of model performance compared to a single accuracy score (Xu et al., 2020).

3.5.2 Sklearn metrics

The classification performance in machine learning can be evaluated using precision, recall, and f1-score, which are already predefined in the sklearn.metrics module (Buitinck et al., 2013). These measures are defined by equations, as shown in the image below (Ćwiklinski et al., 2021).

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The Figure 2. Formula for precision, recall, and f1-score

Precision measures the level of accuracy in true predictions, indicating the proportion of correct predictions out of all positive predictions. It is essential to have high precision (Suresh, 2020).

Recall is a measure of correctly predicted actual observations, indicating the proportion of actual positive results among all observed positive results. A high recall score is desirable (Suresh, 2020).

F1 score, which is a combination of precision and recall, is a useful metric to evaluate model performance. It is calculated as the harmonic mean of precision and recall, and a high f1 score is desirable (Suresh, 2020).

Additionally, it is worth noting the meaning of support, which refers to the number of samples utilized for model training.

Chapter 4. Implementation

4.1 Code repository

The Python Jupyter Notebook file is stored in GitHub repositories and is set to be viewable for public. The link of code repository is shown as below:

https://github.com/HUDongpin/FP/blob/fca79bc73b2e5ad307ad7cdae3c36729f39a0ca7/Final%20Project.ipynb

The relevant HTML file is also attached to the code repository. In case the Jupyter Notebook file is not viewable in some cases, the HTML file can be downloaded and view locally to see how the code part and running results look like. The link of the HTML file is shown as below:

https://github.com/HUDongpin/FP/blob/fca79bc73b2e5ad307ad7cdae3c36729f39a0ca7/Final%20Project.html

4.2 Explanations of code for traditional classification models

Five traditional classification models (Naive Bayes, support vector machine, logistic regression, random forest classifier, and k-nearest neighbors) have common styles of coding. First, I used the predefined class from dependencies to initialise a model object. Second, I fit the model with x training data and y training data. Here x refers to the data on the feature column and y refers to the data on the target column. Third, the trained model object was used to predict the y value based on its x testing data. Finally, the accuracy score of the model was printed out. The code for calculating recall, precision, and f1-score will be explained in *4.4 Explanations of code for displaying confusion matrix* in this chapter.

Now I take Naive Bayes model as an example and show its code and comments for each block of code below.

```
# Create the Multinomial Naive Bayes object and train it

nb_model = MultinomialNB()

nb_model.fit(X_train, y_train)


# Predict the labels of the testing data

y_nb_pred = nb_model.predict(X_test)


# Print the accuracy score
```

```
print("NB Accuracy Score:", accuracy_score(y_test, y_nb_pred))
```

More details about code for other traditional classification models are listed in Appendix I.

4.3 Explanations of code for advanced classification models

Three advanced classification models (LSTM, bidirectional LSTM, and CNN) have common styles of coding. First, I created a Tokenizer object and fit it to the training data to convert the text into sequences. Second, I defined a deep learning model using Keras with an embedding layer, an LSTM layer, and a dense output layer with sigmoid activation. Third, I compiled the model with binary crossentropy loss and Adam optimizer. The value of epochs is set to be 5. Fourth, I trained the model on the training data using the fit() function and validated it on the testing data. Fifth, I also evaluated the model on the testing data to obtain the accuracy score. The model utilized advanced methods in NLP, including word embedding and LSTM layer for sequence modeling. It can provide higher accuracy compared to traditional machine learning models.

Now I take LSTM model as an example and show its code and comments for each block of code below.

```
# Create the tokenizer object and fit it to the text data
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(df['text'])


# Transform the text data into sequences of integers
sequences = tokenizer.texts_to_sequences(df['text'])


# Pad the sequences to have the same length
ltsm_X = pad_sequences(sequences, maxlen=500)


# Create the target variable
ltsm_y = df['whether_real']


# Split the data into training and testing sets
```

```
ltsm_X_train, ltsm_X_test, ltsm_y_train, ltsm_y_test = train_test_split(ltsm_X, ltsm_y,
test_size=0.2, random_state=42)

# Build LSTM model

lstm_model = Sequential()

lstm_model.add(Embedding(10000, 128, input_length=500))

lstm_model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))

lstm_model.add(Dense(1, activation='sigmoid'))

lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model

lstm_model.fit(ltsm_X_train, ltsm_y_train, validation_data=(ltsm_X_test, ltsm_y_test),
epochs=5, batch_size=32)

# Make predictions on the testing set

loss, lstm_accuracy = lstm_model.evaluate(ltsm_X_test, ltsm_y_test, verbose=False)

print('LSTM Accuracy Score:', lstm_accuracy)

# Make predictions on the testing set

y_lstm_pred = np.argmax(lstm_model.predict(ltsm_X_test), axis=1)

print('y_lstm_pred:', y_lstm_pred)
```

It is worth mentioning that the CNN model has two extra layers, namely Conv1D layer and GlobalMaxPooling1D layer to extract important features from the text data. More details about code for advanced classification models are listed in Appendix II.

4.4 Explanations of code for displaying confusion matrix

In order to visually display the confusion matrix for the result of each model, I created a function to fulfill this requirement based on the code from Scikit Learn (2016). This function has six parameters with four default values (i.e., normalize, title, cmap, figure), aiming to create a 2 rows x 2 columns rectangles to display the confusion matrix. The display function for confusion matrix will not have a conflict with sklearn's original matrix display function. The detailed explanations of each code block is illustrated below together with code.

The function named plot_confusion_matrix is shown as follow:

```
from matplotlib import pyplot as plt
```

```
def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix',
cmap=plt.cm.Blues, figure=0):
    """
    Plots the confusion matrix `cm` as a heatmap using matplotlib.

    :param cm: ndarray, confusion matrix
    :param classes: list, class labels
    :param normalize: bool, whether to normalize the confusion matrix or not
    :param title: str, title of the plot
    :param cmap: colormap to use for the heatmap
    :param figure: int, figure number to use for the plot
    """
    # Creating the Matplotlib figure and axis:
    fig, ax = plt.subplots(num=figure)
    # Displaying the confusion matrix as a heatmap:
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    # Setting title and color bar
    ax.set_title(title)
    plt.colorbar(im, ax=ax)
    # Setting tick marks and labels
    tick_marks = np.arange(len(classes))
    ax.set_xticks(tick_marks)
    ax.set_xticklabels(classes, rotation=45)
    ax.set_yticks(tick_marks)
    ax.set_yticklabels(classes)
    # Normalizing the confusion matrix
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    # Adding text annotations to the cells of the heatmap
    thresh = cm.max() / 2.
    for i in range(len(classes)):
        for j in range(len(classes)):
            ax.text(j, i, cm[i, j], ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    # Setting labels for the x and y axes:
    ax.set_ylabel('True label')
    ax.set_xlabel('Predicted label')
    # Displaying the plot
    fig.tight_layout()
    plt.show()
```

Chapter 5. Evaluation

5.1 Evaluation of eight classifiers based on the first dataset

The evaluation of eight classifiers based on the first database have common styles of coding. First, I used confusion_matrix method to calculate the result of confusion matrix for the model. Second, I used the function plot_confusion_matrix to plot the confusion matrix with a title and two classes. Finally, I used classification_report method to print a holistic score report of precision, recall, and f1-score of the model.

5.1.1 Evaluating the Naive Bayes model

Now I take Naive Bayes model as an example and show its code and comments for each block of code below:

```
# Calculate the confusion matrices for the nb model
nb_cm = confusion_matrix(y_test, y_nb_pred)

# Plot the nb_cm confusion matrix
plot_confusion_matrix(nb_cm, classes=['True News', 'Fake News'], title= "Naive Bayes
Confusion Matrix")

# print a holistic report of Naive Bayes classifier report on precision, recall, f1-score, and
support
nb_classifier_report = classification_report(y_test, y_nb_pred)

# print the report
print(nb_classifier_report)
```

The confusion matrix is displayed on Figure 3. Its precision, recall, and f1-score is printed out as 93%, 93%, and 93% respectively.

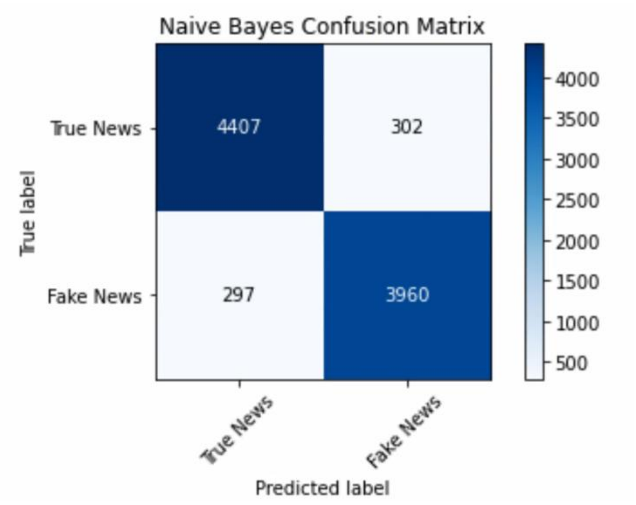

Figure 3. Visualization of confusion matrix for naive Bayes model

As can be seen from Figure 3, the value of true positive is 4407, the value of true negative is 3960, the value of false positive is 297, and the value of false negative is 302. The following part will only list the confusion matrix for each model. More details about code for evaluation of eight classifiers in this study can be found in Appendix III.

5.1.2 Evaluating the Support Vector Machine model

The confusion matrix is displayed on Figure 4:

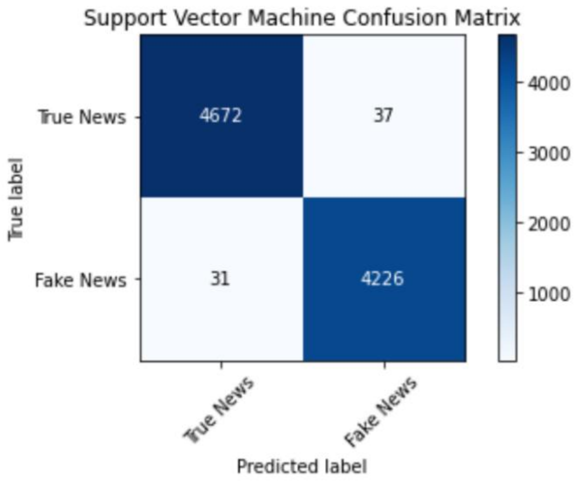

Figure 4. Visualization of confusion matrix for support vector machine model

As can be seen from Figure 4, the value of true positive is 4672, the value of true negative is 4226 , the value of false positive is 31, and the value of false negative of 37. The score report of precision, recall, and f1-score of the support vector machine model is printed out as 99%, 99%, and 99% respectively.

5.1.3 Evaluating the Logistic Regression model

The confusion matrix is displayed on Figure 5:

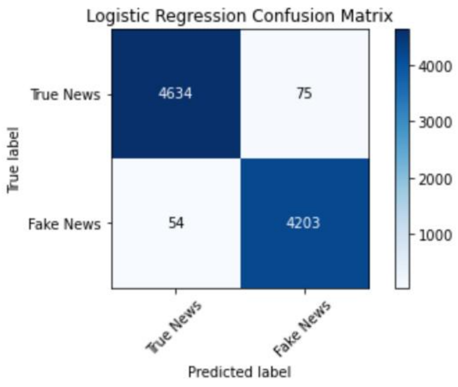

Figure 5. Visualization of confusion matrix for logistic regression model

As can be seen from Figure 5, the value of true positive is 4634, the value of true negative is 4203, the value of false positive is 54 , and the value of false negative of 75. The score report of precision, recall, and f1-score of the logistic regression model is printed out as 99%, 99%, and 99% respectively.

5.1.4 Evaluating the Random Forest model

The confusion matrix is displayed on Figure 6:

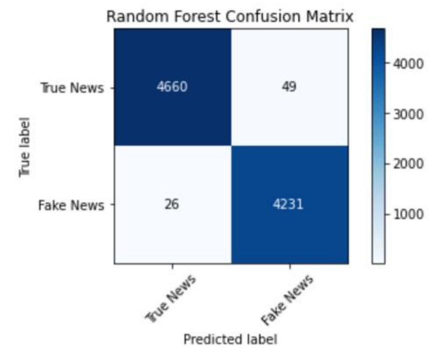

Figure 6. Visualization of confusion matrix for random forest model

As can be seen from Figure 6, the value of true positive is 4660, the value of true negative is 4231, the value of false positive is 26, and the value of false negative of 49. The score report of precision, recall, and f1-score of the random forest model is printed out as 99%, 99%, and 99% respectively.

5.1.5 Evaluating the KNN model

The confusion matrix is displayed on Figure 7:

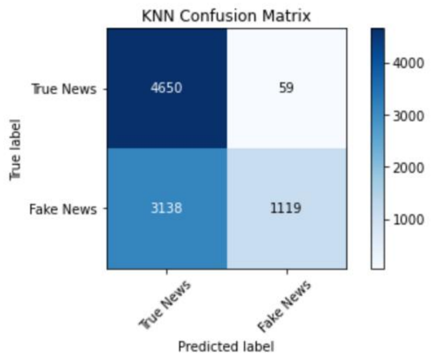

Figure 7. Visualization of confusion matrix for KNN model

As can be seen from Figure 7, the value of true positive is 4650, the value of true negative is 1119, the value of false positive is 3138, and the value of false negative of 59. The score report of precision, recall, and f1-score of the KNN model is printed out as 76%, 64%, and 59% respectively.

5.1.6 Evaluating the LSTM model

The confusion matrix is displayed on Figure 8:

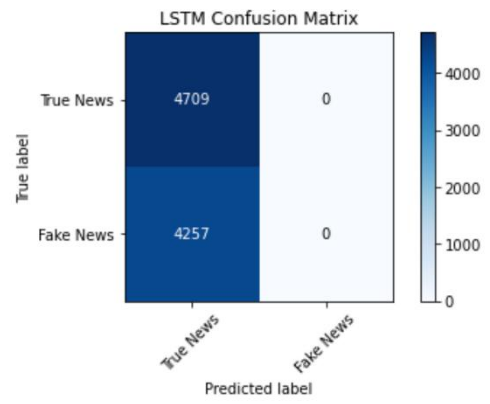

Figure 8. Visualization of confusion matrix for LSTM model

As can be seen from Figure 8, the value of true positive is 4709, the value of true negative is 0, the value of false positive is 4257, and the value of false negative of 0. The score report of precision, recall, and f1-score of the LSTM model is printed out as 28%, 53%, and 36% respectively.

5.1.7 Evaluating the bidirectional LSTM model

The confusion matrix is displayed on Figure 9:

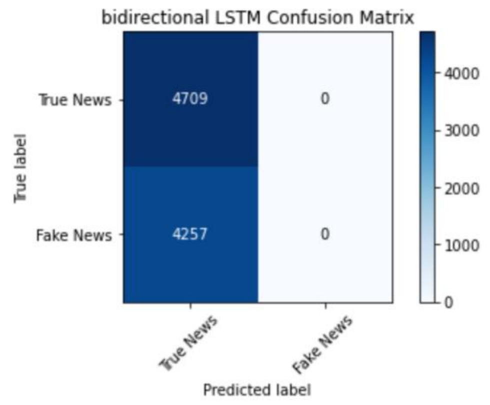

Figure 9. Visualization of confusion matrix for bidirectional LSTM model

As can be seen from Figure 9, the value of true positive is 4709, the value of true negative is 0, the value of false positive is 4257, and the value of false negative of 0. The score report of precision, recall, and f1-score of the bidirectional LSTM model is printed out as 28%, 53%, and 36% respectively.

5.1.8 Evaluating the CNN model

The confusion matrix is displayed on Figure 10:

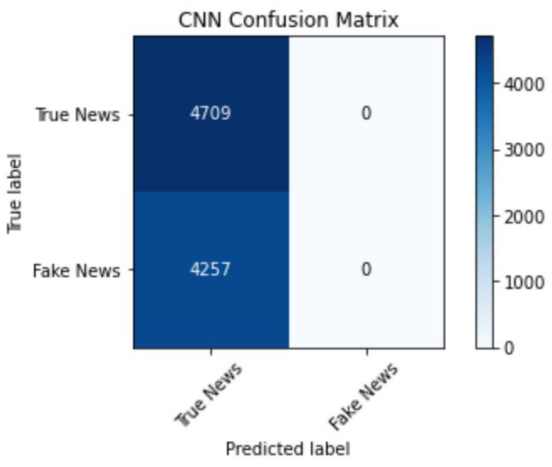

Figure 10. Visualization of confusion matrix for CNN model

As can be seen from Figure 10, the value of true positive is 4709, the value of true negative is 0, the value of false positive is 4257, and the value of false negative of 0. The score report of precision, recall, and f1-score of the CNN model is printed out as 28%, 53%, and 36% respectively.

5.2 Evaluation of eight classifiers based on the second dataset

5.2.1 Evaluating the Naive Bayes model

The confusion matrix is displayed on Figure 11:

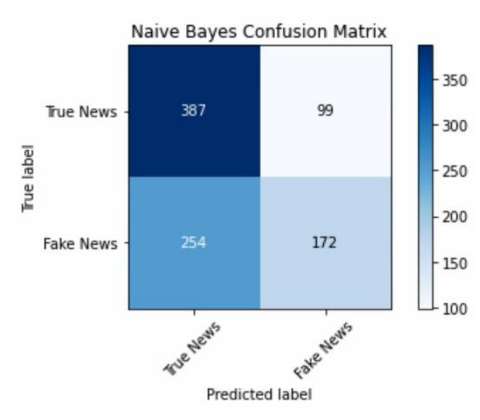

Figure 11. Visualization of confusion matrix for naive Bayes model

As can be seen from Figure 11, the value of true positive is 387, the value of true negative is 172, the value of false positive is 254, and the value of false negative of 99. The score report of precision, recall, and f1-score of the naive Bayes model is printed out as 62%, 61%, and 60% respectively.

5.2.2 Evaluating the Support Vector Machine model

The confusion matrix is displayed on Figure 12:

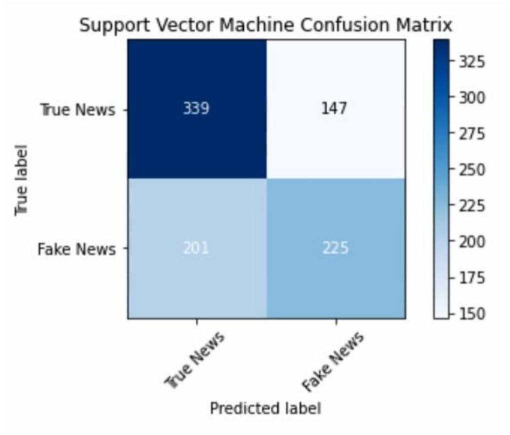

Figure 12. Visualization of confusion matrix for support vector machine model

As can be seen from Figure 12, the value of true positive is 339, the value of true negative is 225, the value of false positive is 201, and the value of false negative of 147. The score report of precision, recall, and f1-score of the support vector machine model is printed out as 62%, 62%, and 62% respectively.

5.2.3 Evaluating the Logistic Regression model

The confusion matrix is displayed on Figure 13:

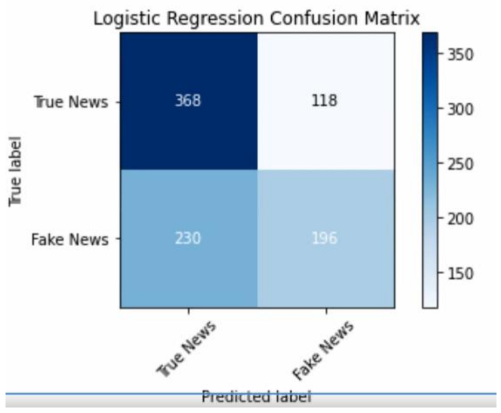

Figure 13. Visualization of confusion matrix for logistic regression model

As can be seen from Figure 13, the value of true positive is 368, the value of true negative is 196, the value of false positive is 230, and the value of false negative of 118. The score report of precision, recall, and f1-score of the logistic regression model is printed out as 62%, 62%, and 61% respectively.

5.2.4 Evaluating the Random Forest model

The confusion matrix is displayed on Figure 14:

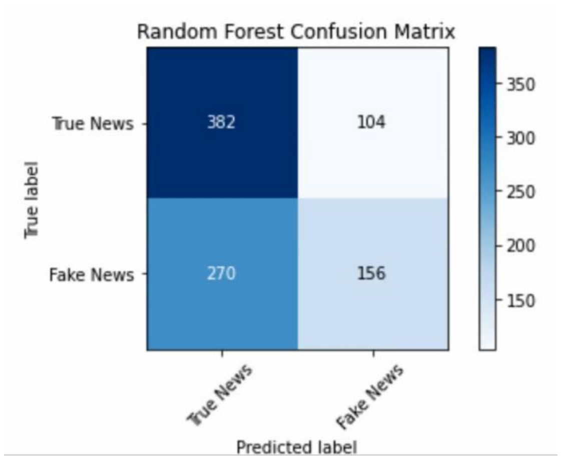

Figure 14. Visualization of confusion matrix for random forest model

As can be seen from Figure 14, the value of true positive is 382, the value of true negative is 156, the value of false positive is 270, and the value of false negative of 104. The score report of precision, recall, and f1-score of the random forest model is printed out as 59%, 59%, and 57% respectively.

5.2.5 Evaluating the KNN model

The confusion matrix is displayed on Figure 15:

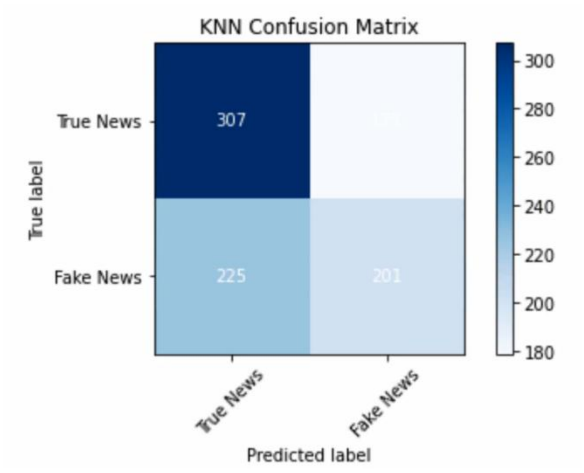

Figure 15. Visualization of confusion matrix for KNN model

As can be seen from Figure 15, the value of true positive is 307, the value of true negative is 201, the value of false positive is 225, and the value of false negative of 179. The score report of precision, recall, and f1-score of the KNN model is printed out as 55%, 56%, and 55% respectively.

5.2.6 Evaluating the LSTM model

The confusion matrix is displayed on Figure 16:

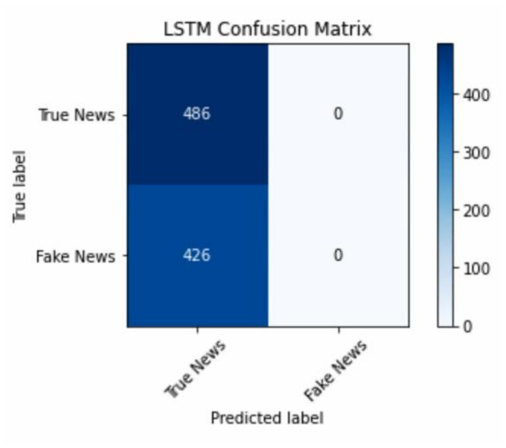

Figure 16. Visualization of confusion matrix for LSTM model

As can be seen from Figure 16, the value of true positive is 486, the value of true negative is 0, the value of false positive is 426, and the value of false negative of 0. The score report of precision, recall, and f1-score of the LSTM model is printed out as 28%, 53%, and 37% respectively.

5.2.7 Evaluating the bidirectional LSTM model

The confusion matrix is displayed on Figure 17:

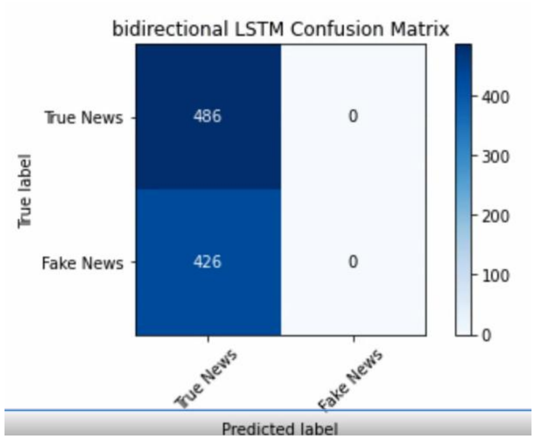

Figure 17. Visualization of confusion matrix for bidirectional LSTM model

As can be seen from Figure 17, the value of true positive is 486, the value of true negative is 0, the value of false positive is 426, and the value of false negative of 0. The score report of precision, recall, and f1-score of the bidirectional LSTM model is printed out as 28%, 53%, and 37% respectively.

5.2.8 Evaluating the CNN model

The confusion matrix is displayed on Figure 18:

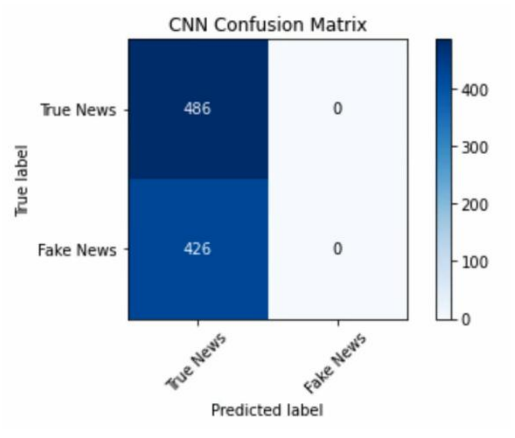

Figure 18. Visualization of confusion matrix for CNN model

As can be seen from Figure 18, the value of true positive is 486, the value of true negative is 0, the value of false positive is 426, and the value of false negative of 0. The score report of precision, recall, and f1-score of the CNN model is printed out as 28%, 53%, and 37% respectively.

Chapter 6. Conclusion

6.1 Summary of the results of models for two datasets

The evaluation results of these eight models used in this project for two datasets are displayed below in a table respectively. In order to keep the table clean, I remove percent sign for each data.

The first table is the summary of evaluation results for the first database, namely ISOT fake news dataset.

Table 2. Performance of eight machine learning methods for fake news detection (first database)

model	accuracy score	precision	recall	f1-score
Naive Bayes	93	93	93	93
Support Vector Machine	99	99	99	99
Logistic regression	98	99	99	99
Random Forest	99	99	99	99
KNN	64	76	64	59
LSTM	98	28	53	36
Bidirectional LSTM	99	28	53	36
CNN	99	28	53	36

Based on the first table, it is obvious that both Random Forest and Support Vector Machine perform the best among these eight models for fake news detection. It is worth noting that Logistic Regression had the same performance to these two models except for the accuracy score being 1% smaller than that of these two models. Among these eight models, Naive Bayes ranks fourth in performance. The performance of KNN model had the lowest accuracy among these eight models and its f1-score is lower than the performance of Random Forest, Support Vector Machine, Logistic Regression, and Naive Bayes. However, the f1-score of KNN model is better than the performance of these three advanced models, namely LSTM, bidirectional LSTM, and CNN.

Surprisingly, the three advanced models, which require relatively larger computational power, perform extremely well in accuracy (99% accuracy for each model), but these three models perform unsatisfactorily in precision, recall, and f1-score, even largely beaten by simple models like Naive Bayes and KNN.

The second table is the summary of evaluation results for the second database, namely LIAR fake news dataset.

Table 3. Performance of eight machine learning methods for fake news detection (second database)

model	accuracy score	precision	recall	f1-score
Naive Bayes	61	62	61	60
Support Vector Machine	62	62	62	62
Logistic regression	62	62	62	61
Random Forest	59	59	59	57
KNN	56	55	56	55
LSTM	60	28	53	37
Bidirectional LSTM	59	28	53	37
CNN	63	28	53	37

Based on the second table, it is obvious that Support Vector Machine had the best performance among these eight models for fake news detection, closely followed by Logistic Regression with only 1% smaller in f1-score. It is worth noting that Naive Bayes had the same performance to these two models except for the accuracy score, recall, and f1-score being 1% smaller than these two models. Among these eight models, Random Forest ranks fourth in performance. The performance of KNN model had the lowest accuracy among these eight models and its f1-score is lower than the performance of Random Forest, Support Vector Machine, Logistic Regression, and Naive Bayes. However, the f1-score of KNN model is better than the performance of these three advanced models, namely LSTM, bidirectional LSTM, and CNN.

The same event happened again. The three advanced models, which require relatively larger computational power, perform extremely well in accuracy (99% accuracy for each model).

Although the three advanced models have a higher performance of accuracy than KNN, and CNN even has the highest accuracy among these eight models, the f1-score of these three advanced models does not achieve a satisfactory performance, even largely beaten by simple models like Naive Bayes and KNN.

6.2 Summary of the project

First, Support Vector Machine performs the best among these eight models for both datasets. Support Vector Machine is usually regarded as an efficient and fast-running classifier for text classification (Mishra et al., 2022). Thus, for personal users and small companies when using machine learning methods for text classification, Support Vector Machine may become a very suitable and cheap choice because it does not require expensive computing power. In addition, Support Vector Machine can generate the text classification result much faster than advanced models like LSTM, bidirectional LSTM, and CNN.

Second, the three traditional machine learning methods for text classification, namely Random Forest, Logistic Regression, and Naive Bayes, are alternative choices for text classification. For both datasets, these three models have comparable performance to the best performance model Support Vector Machine. In addition, these three models are also fast and do not need large computing power.

Third, KNN is the model which ranked in the middle (i.e., the fifth rank) in terms of precision, recall and f1-score among these eight models for both datasets. However, the performance of the KNN model had the lowest accuracy performance among these eight models for both datasets. To be specific, the f1-score of the KNN model is better than the performance of these three advanced models, namely LSTM, bidirectional LSTM, and CNN, but the performance of the KNN model had the lowest accuracy among these eight models and its f1-score is lower than the performance of Random Forest, Support Vector Machine, Logistic Regression, and Naive Bayes.

Fourth, the three advanced models (i.e., LSTM, bidirectional LSTM, and CNN), which require relatively larger computational power, perform extremely well in accuracy (99% accuracy for each model), but these three models perform unsatisfactorily in precision, recall, and f1-score, even largely beaten by simple models like Naive Bayes and KNN. The reason behind the high accuracy and low f1-score is probably due to overfitting (Ying, 2019).

Fifth, the performance of these five traditional models used in this study is better than the performance of the benchmark of these five traditional models mentioned in the baseline part (i.e., the second chapter of this report). This is mainly caused by the adoption of different datasets in this project and studies in the literature. The performance of classifiers for fake news detection may vary based on the adoption of different datasets (Ahmad & Aftab, 2017).

6.3 Implications of the findings

This final project has made comparisons between five common classifiers (i.e., Naive Bayes, support vector machine, logistic regression, random forest classifier, and K-Nearest Neighbors) and three advanced classifiers (i.e., LSTM, bidirectional LSTM, and CNN) in terms of four metrics, namely accuracy, precision, recall, and f1-score.

This study has found that Support Vector Machine rank the highest among these eight models for fake news detection. The model outperforms computation-intensive models like LSTM, bidirectional LSTM, and CNN. Since Support Vector Machine is fast (i.e., the model will finish its running in 1 minute) and requires significantly less computational power, it is reasonable to adopt the model when facing the task of fake news detection for a dataset like 40 thousand rows of news text. Regarding the fast execution time of Support Vector Machine, it is more economical to run the model for fake news detection compared with advanced but computation-intensive models like LSTM, bidirectional LSTM, and CNN. This conclusion is applicable to datasets smaller than 40 thousand rows. It is highly likely that this conclusion is also applicable to datasets with approximately one hundred thousand rows of news text.

Thus, Support Vector Machine has become a suitable and economical solution for automated fake news detection for personal users and small companies of machine learning models. This small model can be trained in significantly less time and has produced a greater performance to state-of-the-art large models (Tan & Le, 2021). Support Vector Machine is such an energy-efficient and effective classifier, which has a greater model performance to that of advanced models.

6.4 Limitations of the study

There are two limitations to this study. First, only two datasets were used for this project. The labeled datasets ISOT and LIAR have been widely used in fake news detection (Wang, 2017). However, there are other possible datasets for the study of fake news detection (Mdepak, 2019). Most of these datasets are raw and need human data labeling. In other words, most of these datasets lack the target column which indicates whether the text in the same row is a piece of fake news or not.

Second, I used restricted computing power in this project. The experimental computer did not have a discrete graphics card. Thus, I set the epoch as a small number being only 5. It is possible to set the epoch number higher to detect whether the situation of overfitting has improved. However, due to the lack of computing power, it becomes nearly impossible to conduct this trial.

6.5 Suggestions for future research

Future research on this topic can be conducted with more labeled datasets and strong computing power to generate more comprehensive and general research results for model comparison between traditional machine learning methods and advanced machine learning models. In addition, it is also meaningful to add other advanced methods like transformer and mixed models into the study and generate the results to compare between state-of-the-art models and traditional machine learning models.

References

Abu Arqoub, O., Abdulateef Elega, A., Efe Özad, B., Dwikat, H., & Adedamola Oloyede, F. (2022). Mapping the scholarship of fake news research: A systematic review. *Journalism Practice, 16*(1), 56-86.

Ahmad, M., & Aftab, S. (2017). Analyzing the performance of SVM for polarity detection with different datasets. *International Journal of Modern Education and Computer Science, 9*(10), 29.

Ahmed, A. A. A., Aljabouh, A., Donepudi, P. K., & Choi, M. S. (2021). Detecting fake news using machine learning: A systematic literature review. arXiv preprint arXiv:2102.04458.

Ahmed H, Traore I, Saad S (2018). Detecting opinion spams and fake news using text classification. *Journal of Security and Privacy, 1*(1), 1-15.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)* (pp. 1-6). IEEE.

Andersen, J., & Søe, S. O. (2020). Communicative actions we live by: The problem with fact-checking, tagging or flagging fake news – the case of Facebook. *European Journal of Communication, 35*(2), 126-139.

Bahad, P., Saxena, P., & Kamal, R. (2019). Fake news detection using bi-directional LSTM-recurrent neural network. *Procedia Computer Science, 165*, 74-82.

Bai, X. (2018, September). Text classification based on LSTM and attention. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)* (pp. 29-32). IEEE.

Busioc, C., Ruseti, S., & Dascalu, M. (2020). A Literature Review of NLP Approaches to Fake News Detection and Their Applicability to Romanian Language News Analysis. *Revista Transilvania, 10*, 65-71.

Granik, M., & Mesyura, V. (2017, May). Fake news detection using naive Bayes classifier. In *2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON)* (pp. 900-903). IEEE.

Cutler, A., Cutler, D.R., Stevens, J.R. (2012). Random Forests. In: *Zhang, C., Ma, Y. (eds) Ensemble Machine Learning* (pp. 157-175). Springer, New York, NY. https://doi.org/10.1007/978-1-4419-9326-7_5

De Beer, D., & Matthee, M. (2021). Approaches to identify fake news: a systematic literature review. *Integrated Science in Digital Age 2020*, 13-22.

Domínguez-Almendros, S., Benítez-Parejo, N., & Gonzalez-Ramirez, A. R. (2011). Logistic regression models. *Allergologia et immunopathologia, 39*(5), 295-305.

Graves, A., Fernández, S., & Schmidhuber, J. (2005). Bidirectional LSTM networks for improved phoneme classification and recognition. In A*rtificial Neural Networks: Formal Models and Their Applications – ICANN 2005: 15th International Conference*, Warsaw, Poland, September 11-15, 2005. Proceedings, Part II 15 (pp. 799-804). Springer Berlin Heidelberg.

Graves, A. (2012). Long short-term memory. Supervised sequence labelling with recurrent neural networks. In *Studies in Computational Intelligence* (Vol. 385).

Springer.

Habernal, I., & Gurevych, I. (2016, August). Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers) (pp. 1589-1599).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

Jakkula, V. (2006). Tutorial on support vector machine (svm). School of EECS, Washington State University, 37(2.5), 3.

Kaur, S., Kumar, P., & Kumaraguru, P. (2020). Automating fake news detection system using multi-level voting model. *Soft Computing, 24*(12), 9049-9069.

Kramer, O. (2013). K-Nearest Neighbors. In: Dimensionality Reduction with Unsupervised Nearest Neighbors. Intelligent Systems Reference Library, vol 51. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38652-7_2

Li, B., Pandey, S., Fang, H., Lyv, Y., Li, J., Chen, J., ... & Ding, C. (2020, August). Ftrans: energy-efficient acceleration of transformers using fpga. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design* (pp. 175-180).

Manzoor, S. I., & Singla, J. (2019, April). Fake news detection using machine learning approaches: A systematic review. In *2019 3rd international conference on trends in electronics and informatics (ICOEI)* (pp. 230-234). IEEE.

Mdepak. (2019). Fake News Detection - Resources. Retrieved May 26, 2023, from https://github.com/mdepak/fake-news-detection-resources

Mhatre, S., & Masurkar, A. (2021, June). A hybrid method for fake news detection using cosine similarity scores. In 2*021 International Conference on Communication information and Computing Technology (ICCICT)* (pp. 1-6). IEEE.

Mishra, S., Shukla, P., & Agarwal, R. (2022). Analyzing machine learning enabled fake news detection techniques for diversified datasets. Wireless Communications and Mobile Computing, 2022, 1-18.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

Pattekari, S. A., & Parveen, A. (2012). Prediction system for heart disease using Naïve Bayes. *International Journal of Advanced Computer and Mathematical Sciences, 3*(3), 290-294.

Pérez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2017). Automatic detection of fake news. arXiv preprint arXiv:1708.07104.

Samadi, M., Mousavian, M., & Momtazi, S. (2021). Deep contextualized text representation and learning for fake news detection. *Information Processing & Management, 58*(6), 102723.

Sansonetti, G., Gasparetti, F., D’aniello, G., & Micarelli, A. (2020). Unreliable users detection in social media: Deep learning techniques for automatic detection. *IEEE Access, 8*, 213154-213167.

Scikit Learn. (2016). Confusion Matrix. Retrieved August 8, 2023, from https://scikit-learn.org/0.18/auto_examples/model_selection/plot_confusion_matrix.html

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

Shu, K., Wang, S., & Liu, H. (2019, January). Beyond news contents: The role of social context for fake news detection. In *Proceedings of the twelfth ACM international conference on web search and data mining* (pp. 312-320).

Sharma, U., Saran, S., & Patil, S. M. (2020). Fake news detection using machine learning algorithms. *International Journal of Creative Research Thoughts*, 8(6), 509-518.

Singh, V., Dasgupta, R., Sonagra, D., Raman, K., & Ghosh, I. (2017, July). Automated fake news detection using linguistic analysis and machine learning. In *International conference on social computing, behavioral-cultural modeling, & prediction and behavior representation in modeling and simulation* (SBP-BRiMS) (pp. 1-3).

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586.

Suresh A. (2020). What is a confusion matrix? Retrieved December 4, 2022, from https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5

Tacchini, E., Ballarin, G., Della Vedova, M. L., Moret, S., & De Alfaro, L. (2017). Some like it hoax: Automated fake news detection in social networks. arXiv preprint arXiv:1704.07506.

University of Victoria. (n.d.). ISOT Fake News Dataset. Retrieved April 27, 2023, from https://onlineacademiccommunity.uvic.ca/isot/wp-content/uploads/sites/7295/2023/02/ISOT_Fake_News_Dataset_ReadMe.pdf

Vapnik, V. (1999). The nature of statistical learning theory. Springer science & business media.

Wang, W. Y. (2017). " liar, liar pants on fire": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648.

Wang, Y., Ma, F., Jin, Z., Yuan, Y., Xun, G., Jha, K., ... & Gao, J. (2018, July). Eann: Event adversarial neural networks for multi-modal fake news detection. In Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining (pp. 849-857).

Xu, J., Zhang, Y., & Miao, D. (2020). Three-way confusion matrix for classification: A measure driven view. *Information Sciences, 507*, 772-794.

Ying, X. (2019, February). An overview of overfitting and its solutions. In Journal of physics: Conference series (Vol. 1168, p. 022022). IOP Publishing.

Zhang, X., & Ghorbani, A. A. (2020). An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management, 57*(2), 102025.

Zhou, X., & Zafarani, R. (2020). A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys, 53*(5), 1-40.

Appendices

Appendix I Detailed code for traditional classification models

Naive Bayes model

```
# Create the Multinomial Naive Bayes object and train it

nb_model = MultinomialNB()

nb_model.fit(X_train, y_train)


# Predict the labels of the testing data

y_nb_pred = nb_model.predict(X_test)


# Print the accuracy score

print("NB Accuracy Score:", accuracy_score(y_test, y_nb_pred))
```

Support Vector Machine model

```
# Train the SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# Make predictions on the testing set
y_svm_pred = svm_model.predict(X_test)

# Evaluate the performance of the model
svm_accuracy = accuracy_score(y_test, y_svm_pred)
print('SVM Accuracy Score:', svm_accuracy)
```

Logistic Regression model

```
# Train the logistic regression model
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Make predictions on the testing set
y_lr_pred = lr_model.predict(X_test)

# Evaluate the performance of the model
lr_accuracy = accuracy_score(y_test, y_lr_pred)
print('LR Accuracy Score:', lr_accuracy)
```

Random Forest model

```
# Train the Random Forest Classifier model
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

# Make predictions on the testing set
y_rf_pred = rf_model.predict(X_test)

# Evaluate the performance of the model
rf_accuracy = accuracy_score(y_test, y_rf_pred)
print('RF Accuracy Score:', rf_accuracy)
```

K-nearest neighbors (KNN) model

```
# Define the KNN model
knn_model = KNeighborsClassifier(n_neighbors=5)

# Train the model on the training data
knn_model.fit(X_train, y_train)

# Make predictions on the testing set
y_knn_pred = knn_model.predict(X_test)

# Evaluate the model on the testing data
knn_accuracy = accuracy_score(y_test,y_knn_pred)
print('KNN Accuracy Score:', knn_accuracy)
```

Appendix II Detailed code for advanced classification models

Long short-term memory networks (LSTM) model

```
# Create the tokenizer object and fit it to the text data
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(df['text'])

# Transform the text data into sequences of integers
sequences = tokenizer.texts_to_sequences(df['text'])

# Pad the sequences to have the same length
ltsm_X = pad_sequences(sequences, maxlen=500)

# Create the target variable
ltsm_y = df['whether_real']

# Split the data into training and testing sets
ltsm_X_train, ltsm_X_test, ltsm_y_train, ltsm_y_test = train_test_split(ltsm_X, ltsm_y,
test_size=0.2, random_state=42)

# Build LSTM model
lstm_model = Sequential()
lstm_model.add(Embedding(10000, 128, input_length=500))
lstm_model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
lstm_model.add(Dense(1, activation='sigmoid'))

lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
# Train the model

lstm_model.fit(ltsm_X_train, ltsm_y_train, validation_data=(ltsm_X_test, ltsm_y_test),
epochs=5, batch_size=32)


# Make predictions on the testing set

loss, lstm_accuracy = lstm_model.evaluate(ltsm_X_test, ltsm_y_test, verbose=False)

print('LSTM Accuracy Score:', lstm_accuracy)


# Make predictions on the testing set

y_lstm_pred = np.argmax(lstm_model.predict(ltsm_X_test), axis=1)

print('y_lstm_pred:', y_lstm_pred)
```

Bidirectional LSTM model

```
# Create the tokenizer object and fit it to the text data

tokenizer = Tokenizer(num_words=10000)

tokenizer.fit_on_texts(df['text'])


# Transform the text data into sequences of integers

sequences = tokenizer.texts_to_sequences(df['text'])


# Pad the sequences to have the same length

bi_ltsm_X = pad_sequences(sequences, maxlen=500)


# Create the target variable

bi_ltsm_y = df['whether_real']
```

```
# Split the data into training and testing sets
bi_ltsm_X_train, bi_ltsm_X_test, bi_ltsm_y_train, bi_ltsm_y_test = train_test_split(bi_ltsm_X,
bi_ltsm_y, test_size=0.2, random_state=42)

# Build the model
bi_ltsm_model = Sequential()
bi_ltsm_model.add(Embedding(10000, 128, input_length=500))
bi_ltsm_model.add(Bidirectional(LSTM(64, return_sequences=True)))
bi_ltsm_model.add(Bidirectional(LSTM(32)))
bi_ltsm_model.add(Dense(1, activation='sigmoid'))

# Compile the model
bi_ltsm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
bi_ltsm_model.fit(bi_ltsm_X_train, bi_ltsm_y_train, epochs=5, batch_size=32,
validation_data=(bi_ltsm_X_test, bi_ltsm_y_test))

# Evaluate the model
loss, bi_ltsm_accuracy = bi_ltsm_model.evaluate(bi_ltsm_X_test, bi_ltsm_y_test,
batch_size=32)
print('Bi-LSTM Accuracy Score:', bi_ltsm_accuracy)

# Make predictions on the testing set
y_bi_lstm_pred = np.argmax(bi_ltsm_model.predict(bi_ltsm_X_test),axis=1)
print('y_bi_lstm_pred:', y_bi_lstm_pred)
```

Convolutional neural network (CNN)

```
# Create the tokenizer object and fit it to the text data
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(df['text'])

# Transform the text data into sequences of integers
sequences = tokenizer.texts_to_sequences(df['text'])

# Pad the sequences to have the same length
cnn_X = pad_sequences(sequences, maxlen=500)

# Create the target variable
cnn_y = df['whether_real']

# Split the data into training and testing sets
cnn_X_train, cnn_X_test, cnn_y_train, cnn_y_test = train_test_split(cnn_X, cnn_y,
test_size=0.2, random_state=42)

# Build the CNN model
cnn_model = Sequential()
cnn_model.add(Embedding(input_dim=10000, output_dim=128, input_length=500))
cnn_model.add(Conv1D(filters=32, kernel_size=5, padding='same', activation='relu'))
cnn_model.add(GlobalMaxPooling1D())
cnn_model.add(Dense(128, activation='relu'))
cnn_model.add(Dropout(0.2))
cnn_model.add(Dense(1, activation='sigmoid'))

# Compile the model
```

```
cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


# Train the model

cnn_model.fit(cnn_X_train, cnn_y_train, epochs=5, batch_size=32,
validation_data=(cnn_X_test, cnn_y_test))


# Evaluate the performance of the model

loss, cnn_accuracy = cnn_model.evaluate(cnn_X_test, cnn_y_test, batch_size=32)

print('CNN Accuracy Score:', cnn_accuracy)


# Make predictions on the testing set

y_cnn_pred = np.argmax(cnn_model.predict(cnn_X_test),axis=1)

print('y_cnn_pred:', y_cnn_pred)
```

Appendix III Detailed code for evaluation of eight classifiers

SVM model

```
# Calculate the confusion matrices for the svm model
svm_cm = confusion_matrix(y_test, y_svm_pred)

# Plot the svm_cm confusion matrix
plot_confusion_matrix(svm_cm, classes=['True News', 'Fake News'], title= "Support Vector Machine Confusion Matrix")

# print a holistic report of support vector machine classifier report on precision, recall, f1-score, and support
svm_classifier_report = classification_report(y_test, y_svm_pred)

# print the report
print(svm_classifier_report)
```

Logistic Regression model

```
# Calculate the confusion matrices for the lr model
lr_cm = confusion_matrix(y_test, y_lr_pred)

# Plot the lr_cm confusion matrix
plot_confusion_matrix(lr_cm, classes=['True News', 'Fake News'], title="Logistic Regression Confusion Matrix")


# print a holistic report of logistic regression classifier report on precision, recall, f1-score, and support
lr_classifier_report = classification_report(y_test, y_lr_pred)

# print the report
print(lr_classifier_report)
```

Random Forest model

```
# Calculate the confusion matrices for the rf model
rf_cm = confusion_matrix(y_test, y_rf_pred)

# Plot the rf_cm confusion matrix
plot_confusion_matrix(rf_cm, classes=['True News', 'Fake News'], title="Random Forest Confusion Matrix")
```

# print a holistic report of random forest classifier report on precision, recall, f1-score, and support
rf_classifier_report = classification_report(y_test, y_rf_pred)

# print the report
print(rf_classifier_report)

KNN model
# Calculate the confusion matrices for the count_nb model
knn_cm = confusion_matrix(y_test, y_knn_pred)

# Plot the knn_cm confusion matrix
plot_confusion_matrix(knn_cm, classes=['True News', 'Fake News'], title="KNN Confusion Matrix")

# print a hoslitic report of knn classifier report on precision, recall, f1-score, and support
knn_classifier_report = classification_report(y_test, y_knn_pred)

# print the report
print(knn_classifier_report)

LSTM model
# Calculate the confusion matrices for the count_nb model
lstm_cm = confusion_matrix(y_test, y_lstm_pred)

# Plot the lstm_cm confusion matrix
plot_confusion_matrix(lstm_cm, classes=['True News', 'Fake News'], title="LSTM Confusion Matrix")

# print a hoslitic report of LSTM classifier report on precision, recall, f1-score, and support
lstm_classifier_report = classification_report(y_test, y_lstm_pred)

# print the report
print(lstm_classifier_report)

Bidirectional LSTM model

```
# Calculate the confusion matrices for the bi-LSTM model
bi_lstm_cm = confusion_matrix(y_test, y_bi_lstm_pred)

# Plot the bi_lstm_cm confusion matrix
plot_confusion_matrix(bi_lstm_cm, classes=['True News', 'Fake News'], title="bidirectional
LSTM Confusion Matrix")


# print a hoslitic report of bi-LSTM classifier report on precision, recall, f1-score, and
support
bi_lstm_classifier_report = classification_report(y_test, y_bi_lstm_pred)

# print the report
print(bi_lstm_classifier_report)
```

CNN model

```
# Calculate the confusion matrices for the cnn model
cnn_cm = confusion_matrix(cnn_y_test, y_cnn_pred)

# Plot the cnn_cm confusion matrix
plot_confusion_matrix(cnn_cm, classes=['True News', 'Fake News'], title="CNN Confusion
Matrix")
```