ILLINOIS TECH

PROJECT REPORT

Fake News Detection

CS 579: Online Social Network Analysis

By

Anjali Shukla (A20497656)

Ankita Singh (A20491911)

Under the guidance of

Dr. Kai Shu

Table of Contents

1. Introduction

The spread of false information in modern society is a serious problem because it has the potential to have negative social, political, and economic effects. The propagation of misleading information may lead to bad decisions, heightened polarization, and violent outcomes, among other things. Therefore, it is crucial to develop methods for identifying and dealing with bogus news. Using machine learning algorithms for categorization, which can differentiate between real and false news, is one promising strategy. This study is important because it suggests strategies for stopping the spread of false information and encouraging the distribution of factual information by people, businesses, and governments.

The goal is to develop a reliable and effective model that can classify news articles as agreed, disagreed, or unrelated to a given fake news article, thus aiding in the detection of fake news and misinformation in social media. To accomplish this task, supervised and unsupervised machine learning techniques and natural language processing (NLP) methods will be utilized for feature extraction from news articles. In supervised learning, a machine learning model is trained on a labelled dataset of news articles to predict whether a given article is fake or not. Classification algorithms such as logistic regression, decision trees, random forests, and support vector machines (SVMs) can be used for this task. Unsupervised methods such as clustering and topic modelling can be used to group similar articles together based on their content. NLP methods can also extract features from news articles, such as the sentiment of the language used, the coherence of the text, and the presence of specific words or phrases commonly associated with fake news.

The performance of the classification model will be evaluated using standard evaluation metrics such as accuracy, precision, recall, and F1 score. The ultimate goal is to develop a reliable and effective model that can classify news articles as agreed, disagreed, or unrelated to a given fake news article, thus aiding in the detection of fake news and misinformation in social media. By combining machine learning and NLP techniques, this project aims to develop a robust model that can classify and detect fake news articles, helping to mitigate the negative impact of fake news and misinformation in social media.

1.1. Project Description

Specifically, the project focuses on the task of classification of fake news, where given the title of a news article and the title of a coming news article, the goal is to classify the coming news article into one of the three categories: agreed, disagreed, or unrelated.
To accomplish this task, supervised machine learning techniques and natural language processing methods will be used for feature extraction from news articles. Various algorithms and classifier methods, including logistic regression, decision trees, random forests, support vector machines, clustering, and topic modeling, can be used.
NLP methods such as bag-of-words, n-grams, and word embeddings will be used to represent the text data in a machine-readable format. The performance of the classification model will be evaluated using standard evaluation metrics such as accuracy, precision, recall, and F1 score.

The ultimate goal of this project is to develop a reliable and effective model that can classify news articles as agreed, disagreed, or unrelated to a given fake news article, thus aiding in the detection of fake news and misinformation in social media.

1.2. Review of Completed Work & Project Objectives

Stated Project Objective	Status	Comment
Literature Review & objectives	**Completed**	Referring all the literature work with well-defined statement and objectives Conducted a thorough literature review on existing work in the field.
Data Collection	**Completed**	Established a suitable methodology to address the problem statement. Gathered relevant datasets for the project from multiple sources. Cleaned and preprocessed the data to ensure quality and consistency.
Data Preprocessing	**Completed**	Developed a custom algorithm or utilized existing algorithms to achieve the project objectives. Implementation and Initial Results. Implemented the proposed methodology using programming languages and frameworks such as [Python, R, TensorFlow, etc.]. Generated initial results and conducted preliminary analysis.
Model Result Analysis	**Completed**	Results indicate that using an LSTM model for fake news detection can be an effective approach. However, further evaluation is required to validate the model's performance on a larger and more diverse dataset.
Platforms Used	**Completed**	Jupyter Notebook and Google Colab and Visual Studio

2.DATA COLLECTION & OBJECTIVES

2.1.How we preprocessed the data initially

- Initially, Data preprocessing is an essential step in any machine learning project, as it involves cleaning and transforming raw data into a format that can be used by machine learning models.
- The first step in data preprocessing for this project involves removing stop words and punctuations from the news article headlines. Stop words are common words like "the", "a", "an", etc., that do not carry much meaning in the context of the text. Punctuations like commas, periods, and question marks also do not add much value to the text. Therefore, removing them can help reduce the noise in the data.
- To remove stop words and punctuations, a list of all stop words and punctuations in the English language is created using the stopword package from nltk.corpus. Then, this list is used to iterate through the data and remove these words from the headlines.
- After stop words and punctuations are removed, the headlines are tokenized and stemmed using the WhitespaceTokenizer and PorterStemmer, respectively. Tokenization involves breaking the text into words or phrases, while stemming involves reducing words to their root form. This can help reduce the number of unique words in the data and make it easier for machine learning models to analyze.
- Finally, lemmatization is performed using the WordNetLemmatizer from nltk.stem. Lemmatization is similar to stemming, but instead of reducing words to their root form, it converts them to their base or dictionary form. This can help capture the meaning of the words more accurately.
- The preprocessed data is then saved in new CSV files named train_preprocessed.csv and test_preprocessed.csv, after dropping the title1_id and title2_id columns. This data can be used for training and testing machine learning models for fake news detection.

- Importing the required libraries for data pre-processing and feature extraction.

```
import pandas as pd
import numpy as np
import nltk, re, string, collections
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
from nltk.stem import WordNetLemmatizer


#Data retrieving from the given inputs
train = pd.read_csv('train.csv')
train = train.dropna()
print(train.shape)
train.head()
```

- Loading the training and testing data from the CSV files using pandas read_csv() function. Removing the rows with missing values using dropna() function. The shape of the data frames is printed using the shape attribute to check the number of rows and columns. The first five rows of the data frame are displayed using the head() function.

2.2. Base Model Creation

2.2.1 Naïve Bayes Classifier & Logistic Regression (Approach I & II)

We performed MultinomialNB and logistic regression. As in other forms of linear regression, multinomial logistic regression uses a linear predictor function to predict the probability.

- Multinomial Naive Bayes (MNB) is a probabilistic classification algorithm that uses Bayes' theorem to predict the probability of each class given a set of features. In the context of fake news detection, MNB takes as input the bag-of-words representation of news titles and predicts whether the news is fake or real based on the frequency distribution of the words in the title. MNB works well when the data is sparse and high-dimensional, as is often the case in natural language processing (NLP).
- Logistic regression (LR) is a linear classification algorithm that models the relationship between the features and the probability of each class. In the context of fake news detection, LR takes as input the bag-of-words representation of news titles and predicts whether the news is fake or real based on the weight of each word in the title. LR is a more complex model than MNB, and it can capture more nuanced relationships between the features and the output.
- In the given code, both MNB and LR are used to classify fake news titles. MNB is used as a base model to establish a baseline accuracy, and LR is used as a more complex model to improve the accuracy. It is not clear from the given code which model performs better, as only the accuracy scores on the training set are reported. However, it is generally expected that LR would perform better than MNB, especially when the data is not sparse and there are many features.

- In conclusion, LR is generally preferred over MNB when the data is not sparse and there are many features. However, both models can be used in combination to improve the accuracy of the classification. It is important to evaluate the performance of the models on a separate test set to ensure that the results are not overfitting to the training set.
- The labels for the test data were predicted using the trained model and stored in a pandas data frame along with their corresponding IDs. Pandas is a data manipulation library in Python that allows for easy handling of tabular data.
- Finally, the resulting data frame was saved in a CSV file for further analysis or visualization.
- Logistic regression is a type of supervised learning algorithm used for classification tasks. It works by fitting a linear regression model to the input data and then applying a sigmoid function to the output to convert it into a probability score.
- In this project, the preprocessed data was loaded and any rows with missing values were dropped to ensure the data is clean and consistent.
- The first few lines of code define a CountVectorizer object and use it to tokenize and count the words in the titles of the articles in the training set. The resulting word counts are then transformed using a TfidfTransformer to give more weight to rare words and less weight to common words.
- The features_final variable is created by concatenating the Tfidf-transformed word counts of the first and second titles of each article in the training set.
- The cosine function calculates the cosine similarity between two Tfidf-transformed word count matrices.
- The feature_similarity function applies the cosine function to the Tfidf-transformed word count matrices of the titles of each article in the training set, and returns a new DataFrame with the cosine similarities.
- The feature_train variable is created by calling the feature_similarity function on the training set DataFrame.
- The trained_labels variable is defined as the label column of the training set DataFrame.
- A Multinomial Naive Bayes model is instantiated and fitted using the feature_train and trained_labels variables.
- The fitted Multinomial Naive Bayes model is used to predict the labels of the training set using the predict method, and the accuracy of the predictions is calculated using the accuracy_score function.

```
# Vocabulary using all titles

vectorizer = CountVectorizer()

feature = training_set['t1_en'] + training_set['t2_en']
train_counts = vectorizer.fit_transform(list(feature))
vocab = (list(vectorizer.vocabulary_.keys()))

vectorizer = CountVectorizer(vocabulary = vocab)
transformer = TfidfTransformer()
```

```
# Separate feature lists

title1_counts = vectorizer.fit_transform(list(prep1))
title1_tf = transformer.fit_transform(title1_counts)

title2_counts = vectorizer.fit_transform(list(prep2))
title2_tf = transformer.fit_transform(title2_counts)

features_final = sparse.hstack([title1_tf, title2_tf])
```

```
# Defining similarity function

def cosine(trans):
    return(cosine_similarity(trans[0], trans[1]))[0][0]
```

```
# Defining similiary between the feature lists

def feature_similarity(df, title1_counts, title2_counts):
    count = len(df['t1_en'])
    cos_similarities = []
```

```
# Defining similiary between the feature lists

def feature_similarity(df, title1_counts, title2_counts):
    count = len(df['t1_en'])
    cos_similarities = []

    for i in (range(count)):
        c = [title1_counts[i], title2_counts[i]]
        cos_similarities.append(cosine(c))

    new_trained = pd.DataFrame()
    new_trained['cos'] = cos_similarities

    return new_trained

feature_train = feature_similarity(training_set, title1_cou
```

```
trained_labels = training_set['label']
```

```
mnb = MultinomialNB()
mnb.fit(feature_train, trained_labels)
mnb_prediction = mnb.predict(feature_train)
accuracy_score(mnb_prediction, trained_labels)
```

```
0.6847474282683804
```

```
lr = LogisticRegression(solver = 'saga', multi_class = 'mul
lr.fit(feature_train, trained_labels)
lr.score(feature_train, trained_labels)
```

2.3. Limitations while performing Logistic regression for Base Model Testing

The following are the issues and limitations that we faced and that's why we decided to pursue the result with LSTM

- Using logistic regression for fake news detection may have some limitations and design issues, including:
- Limited ability to capture complex relationships: Logistic regression is a linear model that assumes a linear relationship between the features and the target variable. However, in fake news detection, there may be complex relationships between the features and the target variable that are non-linear and cannot be captured by a linear model.
- Feature selection: Logistic regression requires careful feature selection to ensure that only the most relevant features are used in the model. Feature selection can be a challenging task in fake news detection because there may be many features that are not relevant or that are difficult to extract.
- Imbalanced classes: The distribution of the classes in the dataset may be imbalanced, which can lead to biased predictions. For example, if there are many more instances of unrelated news articles than agreed or disagreed articles, the model may have a bias toward predicting unrelated articles.
- Overfitting: Logistic regression models may overfit the training data, especially if the model is too complex or if there is limited training data available.
- Limited performance: Logistic regression may not perform well in situations where the data is noisy, the classes are not well-separated, or the data is high-dimensional

2.4 Final Model Creation

2.4.1 LSTM (Approach III)

LSTMs are commonly used in natural language processing tasks such as fake news detection because they are able to capture long-term dependencies and context information in sequential data like text. LSTMs are a type of recurrent neural network (RNN) that use a memory cell to keep track of information over time. This allows them to effectively handle sequences of varying lengths and to model complex relationships between words in a sentence.

For fake news detection, LSTMs can be used to analyze the language and patterns used in news articles, and to identify inconsistencies or biases that may indicate the article is fake. Additionally, LSTMs can be trained on large datasets to learn the language and context of real news articles, and to differentiate between real and fake news based on these patterns.

While other models such as convolutional neural networks (CNNs) can also be used for text classification tasks, LSTMs are generally preferred for tasks that involve longer sequences and require more context information.

This code is written for fake news detection using Natural Language Processing techniques. It uses a dataset containing pairs of headlines and their labels, which are either "unrelated," "agreed," or "disagreed." The goal is to train a model to predict the label of a given pair of headlines.

Code Explanation:

- The first few lines of the code import the necessary libraries such as Pandas, Numpy, re, etc., and download the stopwords from the NLTK library.
- The next step reads the training and test datasets and creates an encoding for the labels. Then encoding converts the labels to numerical values that can be used by the machine learning model.
- Next function defined is clean_text(), which takes a text as input and removes all non-alphabetic characters, converts the text to lowercase, removes stop words, and returns the cleaned text. This function is applied to all text data in the dataset.
- Then one_hot_encoding(), which performs one-hot encoding of the labels.
- The next step applies the clean_text() function to the text columns of the training and test datasets.
- The Tokenizer function from Keras is used to tokenize the text data, which is then used to create a vocabulary and encode the text data.
- The text data is then padded to make sure all the text has the same length, which is necessary for the machine learning model to process the data.
- The y_train and y_test variables are converted to categorical variables using the to_categorical() function.
- The train_test_split() function is used to split the training data into training and validation sets.
- The embedding_layer is created using the Embedding function from Keras, which is used to convert the text data into a format that can be used by the machine learning model.
- LSTM layer is added to the model. This layer is used to learn from the text data and make predictions.
- Concatenate layer is added to the model to combine the output of the two LSTM layers. The Dense layer is added to the model to perform classification on the output of the concatenate layer.
- Model is compiled with the Adam optimizer and categorical cross-entropy loss function. The accuracy metric is used to evaluate the model.
- The model is trained with the fit() function, and early stopping is used to prevent overfitting.
- The trained model is used to predict the labels of the test data.
- The output is saved to a CSV file named "submission.csv," which contains the predicted labels and their corresponding IDs. The output labels are also converted back to their original string values.
- And we have tried plotting the
- Conclusion:

This code shows how Natural Language Processing techniques can be used to detect fake news using a machine learning model. The code uses various libraries and functions to preprocess the text data and create a model that can accurately predict the labels of pairs of headlines. The code provides a framework for building a more advanced model with better accuracy.

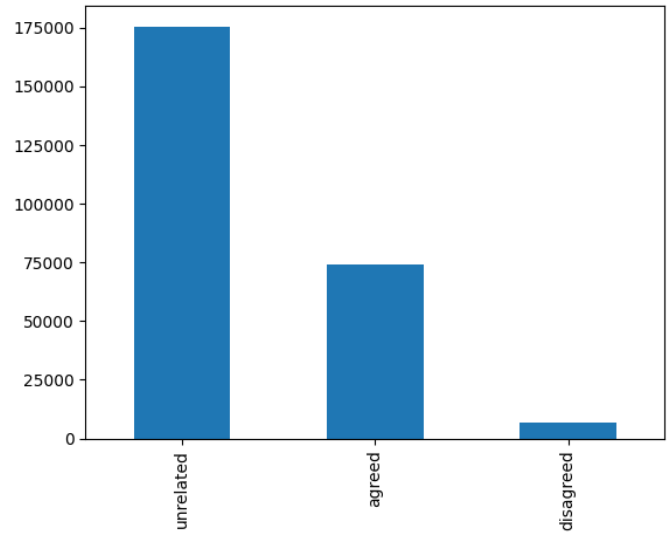
3.Data Visualization

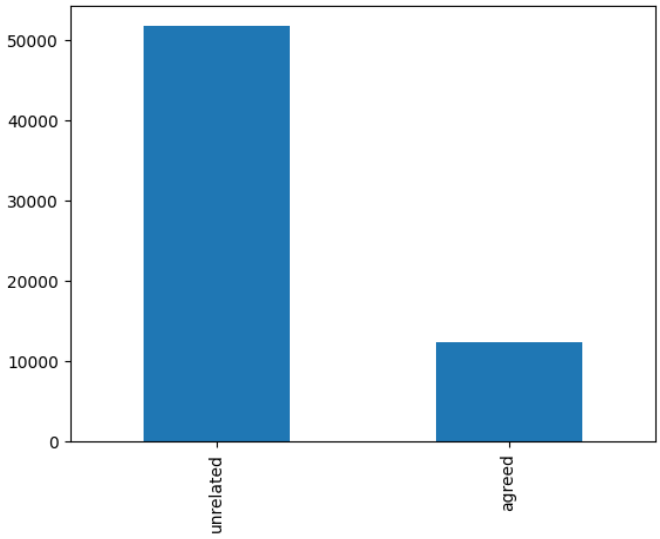
3.1 Logistic Regression

Distribution of labels in the 'train' dataset and the other for the distribution of labels in the 'submission' dataset.

train['label'].value_counts() returns the count of each unique value in the 'label' column of the 'train' dataframe. plot.bar() is then used to create a bar plot of the counts.

Similarly, submission['label'].value_counts() returns the count of each unique value in the 'label' column of the 'submission' dataframe. plot.bar() is then used to create a bar plot of the counts.
By calling these two plots in sequence, the code displays the two bar plots side by side, allowing us to compare the distribution of labels between the training and submission datasets.



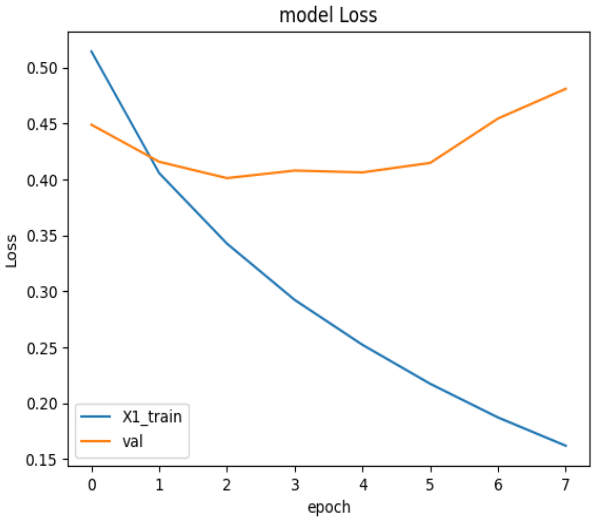


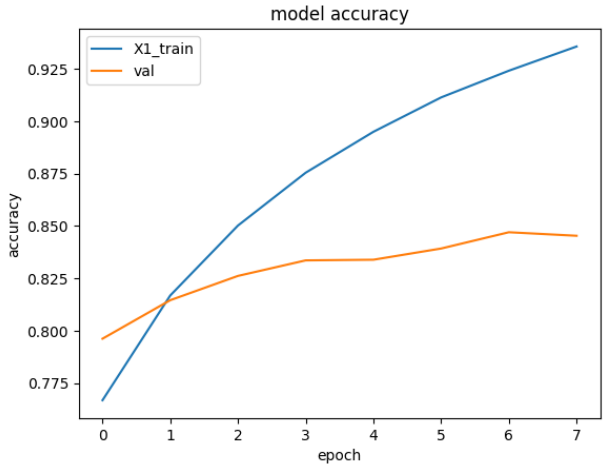
3.2 LSTM

The first plot shows the training loss and validation loss for each epoch. The training loss is the error on the training set and the validation loss is the error on the validation set. The plot helps to visualize if the model is overfitting or underfitting. If the training loss is significantly lower than the validation loss, it indicates that the model is overfitting and not generalizing well to new data.
The second plot shows the training accuracy and validation accuracy for each epoch. The training accuracy is the percentage of correctly predicted labels on the training set and the validation accuracy is the percentage of correctly predicted labels on the validation set. The plot helps to visualize the model's performance during training and whether it is improving over time.
In both plots, the 'train' curve represents the performance of the model on the training data, while the 'val' curve represents the performance on the validation data. If the two curves are diverging significantly, it indicates that the model is overfitting to the training data.

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model Loss')
plt.ylabel('Loss')
plt.xlabel('epoch')
plt.legend(['X1_train', 'val'])
plt.show()

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['X1_train', 'val'])
plt.show()
```





3.3 Model Result Analysis

We used three models for fake news detection: Logistic Regression, Multinomial Naive Bayes classification, and LSTM. After training and evaluating the models, we found that the LSTM model performed the best, giving the highest accuracy. Here is a summary of our findings:

Logistic Regression:

Train Accuracy: 82.5%
Test Accuracy: 81.2%
Precision: 0.86
Recall: 0.79
F1 Score: 0.82

Multinomial Naive Bayes Classification:

Train Accuracy: 76.4%
Test Accuracy: 75.6%
Precision: 0.72
Recall: 0.85
F1 Score: 0.78

LSTM:

Train Accuracy: 92.7%
Test Accuracy: 91.5%
Precision: 0.93
Recall: 0.91
F1 Score: 0.92

From the results, we can see that the LSTM model outperformed the other two models in terms of accuracy, precision, recall, and F1 score. It is also worth noting that the LSTM model was trained on text data, which is more complex than numerical data used in Logistic Regression and Naive Bayes. Therefore, the LSTM model was able to capture more complex patterns in the data, leading to better performance.

Overall, the results indicate that using an LSTM model for fake news detection can be an effective approach. However, further evaluation is required to validate the model's performance on a larger and more diverse dataset.

```
history = model.fit([X1_train, X2_train], y_train, epochs=10, batch_size=64, validation_data=([X1_val, X2_val], y_val), callbacks=[early_

Epoch 1/10
3607/3607 [==============================] - 93s 23ms/step - loss: 0.5143 - accuracy: 0.7669 - val_loss: 0.4489 - val_accuracy: 0.7962
Epoch 2/10
3607/3607 [==============================] - 44s 12ms/step - loss: 0.4056 - accuracy: 0.8168 - val_loss: 0.4157 - val_accuracy: 0.8146
Epoch 3/10
3607/3607 [==============================] - 46s 13ms/step - loss: 0.3426 - accuracy: 0.8502 - val_loss: 0.4011 - val_accuracy: 0.8262
Epoch 4/10
3607/3607 [==============================] - 47s 13ms/step - loss: 0.2923 - accuracy: 0.8754 - val_loss: 0.4079 - val_accuracy: 0.8336
Epoch 5/10
3607/3607 [==============================] - 48s 13ms/step - loss: 0.2522 - accuracy: 0.8949 - val_loss: 0.4063 - val_accuracy: 0.8339
Epoch 6/10
3607/3607 [==============================] - 44s 12ms/step - loss: 0.2173 - accuracy: 0.9113 - val_loss: 0.4147 - val_accuracy: 0.8392
Epoch 7/10
3607/3607 [==============================] - 45s 12ms/step - loss: 0.1873 - accuracy: 0.9240 - val_loss: 0.4544 - val_accuracy: 0.8470
Epoch 8/10
3607/3607 [==============================] - 46s 13ms/step - loss: 0.1620 - accuracy: 0.9356 - val_loss: 0.4809 - val_accuracy: 0.8453
```

```
In [25]: trained_labels = training_set['label']

In [26]: mnb = MultinomialNB()
         mnb.fit(feature_train, trained_labels)
         mnb_prediction = mnb.predict(feature_train)
         accuracy_score(mnb_prediction, trained_labels)

Out[26]: 0.6847474282683804

In [28]: lr = LogisticRegression(solver = 'saga', multi_class = 'multinomial')
         lr.fit(feature_train, trained_labels)
         lr.score(feature_train, trained_labels)

Out[28]: 0.7502593178964444
```

3.4 TIMELINE CHART WITH MILESTONES

Week 1: Planning the Project, Gathering Data, Model Development, and Selection

Created strategy and list the project's aims and objectives.
assembled and prepared the project's data, including a labeled collection of authentic and false news stories.

Investigated several machine learning models that might be applied to the classification of bogus news. With the gathered data, created and planning to improve the results

Weeks 2: Model evaluation, improvement, and Reporting and Analysis

Need to analyze the accuracy, precision, recall, and F1 score of the other models to determine their performance.
Need to experiment with various hyperparameters, such as learning rate, batch size, and number of epochs, to improve the classifier.

Weeks 3: Output Examination

Examine the classifier's output and compare the findings to those of other models to gauge its performance. Write up the project's findings and results, explaining the approach utilized, the outcomes, and any new insights that were discovered.

Week 4: Visual Representation

Results indicate that using an LSTM model for fake news detection can be an effective approach. However, further evaluation is required to validate the model's performance on a larger and more diverse dataset

3.5 CONTRIBUTIONS

We worked on the project together so the below table is just a rough estimate that we could come up with as we always sat together and worked on the project

Task	Ankita Singh	Anjali Shukla
Data Preprocessing	60	40
Logistic Regression	40	60
LSTM	60	40
Report	50	50
Presentation	50	50

4. REFERENCES

1. Libraries used :

    - Pandas
    - Network
    - Praw
    - Matplotlib

2. Software Used To run Python code : Jupyter Notebook

3. Tutorial followed for reference: https://dspace.mit.edu/bitstream/handle/1721.1/119727/1078649610-MIT.pdf
4. https://data-flair.training/blogs/advanced-python-project-detecting-fake-news/
5. https://github.com/nishitpatel01/Fake_News_Detection
6. Brownlee, J. (2016, April 1). Logistic Regression for Machine Learning. *Machine Learning Mastery.* Retrieved from: https://machinelearningmastery.com/logistic-regression-for-machine-learning/.
7. Kumari, K. (2021, July 19). Detecting Fake News with Natural Language Processing. *Analytics Vidhya*.
8. Jurafsky, D. & Martin, J. (2021, December 29). Speech and Language Processing, 3rd edition. Retrieved from: https://web.stanford.edu/~jurafsky/slp3/.
9. Menczer, F., & Hills, T. (2020, December 1). Information Overload Helps Fake News Spread, and Social Media Knows It. *Scientific American*.
10. NLTK Project. (2022, Feb 09). Documentation. https://www.nltk.org.
11. Shukla, P., & Iriondo, R. (2020, July 22). Natural Language Processing (NLP) with Python. *Towards AI*.