Semester Project: Stance Detection

Sripal reddy Nomula, Avinash Reddy Mettukuru, Vidushi Bhargava, Divya Jagabattula

Abstract
The proliferation of fake news and clickbait in recent years has raised serious concerns about its detrimental impact on individuals and society. Hence, there is an urgent need to detect clickbait(misleading headlines), and fake news with reliable and efficient methods. In this project, we propose to develop machine-learning models for clickbait classification using stance detection and fake news detection using datasets containing articles with titles, bodies and relations between them, and real and fake news articles respectively. We will leverage natural language processing techniques to extract key features from the text and build a classifier to accurately distinguish between real and fake news articles. Our goal is to develop a model utilizing stance analysis, which can effectively classify the relation between titles and bodies, and another model for detecting news as fake or real. We plan to utilize the categorization of the provided news into political news, world news, common news, etc. After successfully developing a well-trained and tested model our final goal is to deploy the application onto the web for the public and develop APIs so that other developers can integrate our system as a service. We plan to develop multiple models and thoroughly test them using industry-standard metrics such as F1-score, AUC-ROC, precision, recall etc., and select a single model or an ensemble of them, whichever does best.

Keywords
Stance Detection — Fake news detection — NLP — Ensemble Model — Sentiment analysis — word cloud — Data mining — TFIDF — vectorization — Exploratory data analysis

[1] *Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*

Contents

1. Problem and Data Description

The propagation of misleading information that can be harmful and the waste of people's time makes fake news a major issue on the internet. To draw viewers in, it employs exaggerated headlines that hold little relation to the text's actual content. People are less likely to believe the information they find online as a result. This has an impact on both people and businesses/organizations that rely on online advertising.

Distinguishing between real news and fake news can be challenging since fake news is often designed to look like real news. Fake news can take many forms, but we will focus on two main types: deliberately spreading false or misleading articles and luring people with clickbait(stance detection) that doesn't accurately reflect the content of the article. Traditional methods of detecting fake news using language or topic-based approaches can be unreliable as they may only look at the overall tone of the content and can be deceived if the title and content have a similar tone. Hence, it is essential to use more advanced and dependable techniques to detect fake news accurately.

With recent advancements in data and machine learning algorithms, sentiment analysis has emerged as a more accurate and effective technique for fake news detection. Sentiment analysis is the process of analyzing text data to determine the emotional tone, attitude, or opinion expressed by the writer or speaker. It is also known as opinion mining or emotion AI. The goal of sentiment analysis is to identify and extract subjective information from text, such as the writer's opinion,

feelings, emotions, and attitudes towards a particular subject, product, service, or event. As fake news is a type of misinformation that is intentionally spread to mislead people or to generate clicks, views, or engagement on social media platforms. However, it is important to note that sentiment analysis alone may not be sufficient for detecting fake news. Other techniques such as fact-checking, source verification, and content analysis may also be needed to verify the accuracy of the news article or post, these additional techniques will be represented later on in this project. By detecting and filtering out fake news, the internet can become a more trustworthy and valuable source of information for everyone.

The goal of this project is to create an accurate and efficient model that can quickly identify fake news content in two ways one is true/fake classification and the other is stance detection. Using natural language processing, this model should be able to analyze and sort through a lot of text data and extract pertinent information. The project's output will be a useful tool for people, companies, and organizations to recognize and avoid fake news content, thereby minimizing wastage of time and potential injury and boosting confidence in online information sources.

We have selected two CSV files for true or fake classification, containing news articles related to politics and world news, published between January 13, 2016, and December 31, 2017. The True.csv dataset includes four main columns: title, text, subject, and date. A sample data can be seen in Figure:[1]. The "title" column has 20,826 unique values, all of which are valid. The "text" column has 21,192 unique values, all of which are valid. The "subject" column contains 21,400 valid entries and has two categories: "politicsNews" or "world news". The "date" column contains 21,400 valid entries and ranges from January 13, 2016, to December 31, 2017. The mean date is June 3, 2017, and the period from September 14, 2017, to October 20, 2017, had the most articles, with 3,806 publications. There are no missing or mismatched values in this dataset. on the other hand, The "Fake" dataset provides

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017

Figure 1. Sample of the dataset

information about the publishing date, subject, title, and text. While the "subject" column distinguishes items as "news," "politics," or "other," the "title" column has 17,903 distinct and valid entries. The dataset contains a total of 23.5k valid items with no missing values; however, the "date" column contains 10 mismatched values. The publishing date ranges from March 30, 2015, to February 18, 2018, with March 30, 2015, being the minimum, October 7, 2016, being average, and February 18, 2018, being the maximum.

The Stance Training dataset as shown in figure[2] contains 49972 instances and the Test dataset contains 25413 instances. Each instance consists of a headline, a body of text, and a

	Body ID	articleBody	Headline	Stance
0	0	A small meteorite crashed into a wooded area i...	Soldier shot, Parliament locked down after gun...	unrelated
1	0	A small meteorite crashed into a wooded area i...	Tourist dubbed □Spider Man□ after spider burro...	unrelated
2	0	A small meteorite crashed into a wooded area i...	Luke Somers 'killed in failed rescue attempt i...	unrelated
3	0	A small meteorite crashed into a wooded area i...	BREAKING: Soldier shot at War Memorial in Ottawa	unrelated
4	0	A small meteorite crashed into a wooded area i...	Giant 8ft 9in catfish weighing 19 stone caught...	unrelated

Figure 2. Sample of the stance dataset

stance label which can be one of four categories: unrelated, discuss, agree, or disagree. The dataset does not have any missing values but has a small percentage (1%) of duplicate values that were ignored during the model training process. However, the dataset is highly biased with respect to the target class, meaning that certain stance labels are over-represented while others are underrepresented.

2. Data Pre-processing & Exploratory Data Analysis

Data pre-processing and exploratory data analysis (EDA) are crucial steps in any data analysis project. These steps involve cleaning, transforming, and analyzing the data to ensure its quality and suitability for modeling.

For the True and Fake data sets, the following pre-processing and EDA steps have be taken:

2.1 Handling Missing Values

We thoroughly checked our training data set in the data pre-processing stage for duplicate rows or columns. Numerous hundred instances of duplicate values were found, which may have a detrimental effect on the precision of our predictions. So that our outcomes wouldn't be impacted, we eliminated these duplicates.

Since only four variables in our data set—Title, Text, Subject, and Date stamp—were crucial for determining whether an article is in agreement or disagreement, we did not deliberately remove any of these columns. Instead, we eliminated any extra columns we had added during data separation that did not benefit our research.

We carefully examined our data set for any missing or wrong data values and rectified them as necessary to assure the accuracy and dependability of our data set for exploratory data analysis. This careful method enabled us to create a reliable model based on precise data, which in turn increased forecast accuracy.

2.2 Exploratory Data Analysis

In order to conduct exploratory data analysis on our data set for the train data set, the columns available for analysis are
Title: this is the headlines of the articles
Text: Body of the article
Subject: category of the article(sports, political etc.,)
date stamp: Date on which the article was published.

label: The label column categorizes each news article as either reliable or potentially unreliable, with a value of 0 indicating a reliable article and a value of 1 indicating a potentially unreliable article. The Figure:[3] explain each of these variables:

Field	Datatype	Description
title	object	Headline of the article
text	object	Body of the article
subject	object	News/ politics/left-news /Government News/US_News/Middle-east
date	dateTime	date when the post was published

Figure 3. Dataset individual attribute description

To begin the EDA, the first step is to analyze the distribution of the label column. This will provide insight into the balance of reliable and potentially unreliable articles in the dataset.

The plots show the distribution of the ”class” column in the dataset shown in Figure:[4] and Figure:[5], which indicates whether an article is titled ”True” or ”Fake” with the body. Figure:[4] shows a pie chart representing the percentage of articles in each class. It shows that around 56.3% of the articles are ”Fake” and around 43.7% of the articles are ”True”.

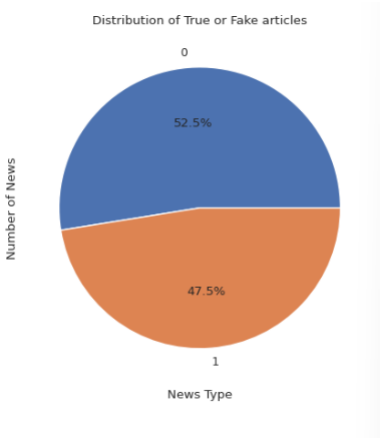

Figure 4. Distribution of True or Fake articles using Pie-chart

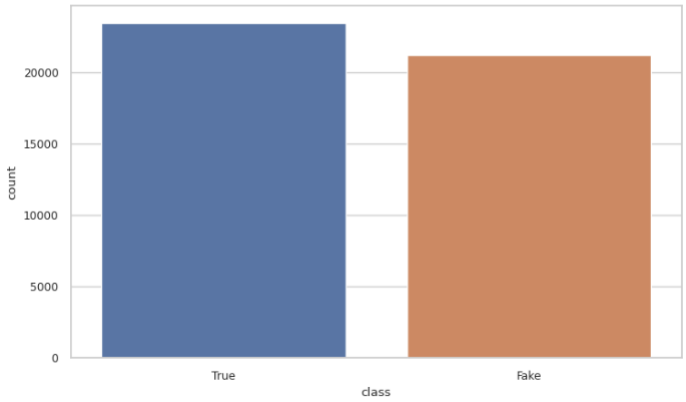

Figure 5. Distribution of True or Fake articles using bar plot

Figure:[5] shows a bar chart with the number of articles in each class. It shows that there are more ”Fake” articles than ”True” articles, with a difference of around 500 articles. Overall, the plot suggests that the dataset is slightly imbalanced towards the ”Fake” class, and this should be taken into consideration when building a machine-learning model. The overall distribution of the News in various categories is shown in the Figure:[6] .It can be observed from the pie chart that the maximum spread of fake news is in the category politics news (25.2%) followed by world news (22.6%) followed by general News(20.2%) and then politics(15.2%) and a very minor percentage of government, US, Middle-east categories.

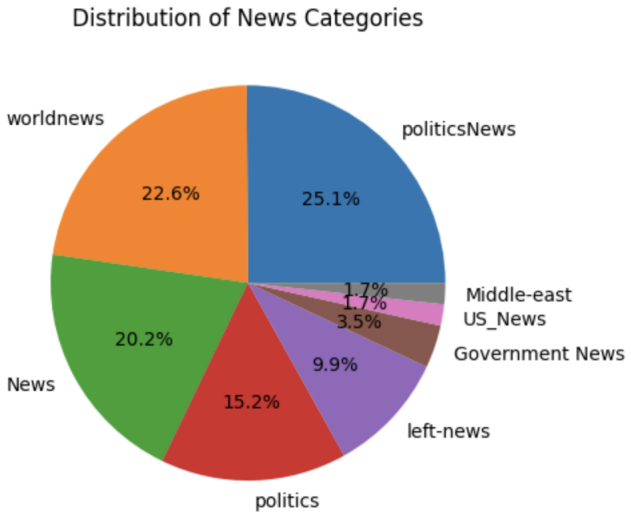

Figure 6. Overall distribution of news by category

The plots in Figure:[7], Figure:[8] displayed depict the distribution of true and fake articles over a period of time from March 2015 to February 2018.

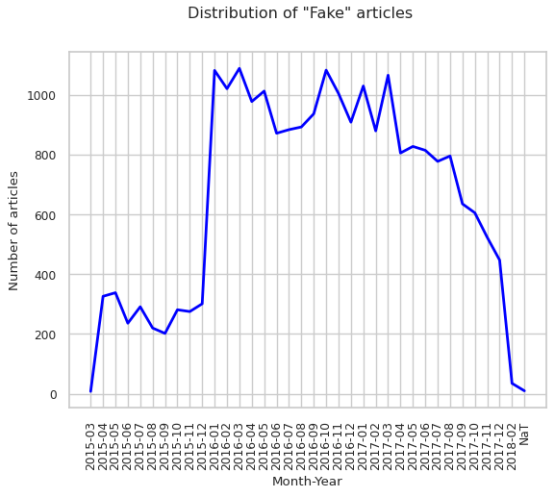

Figure 7. Distribution of Fake Articles by month and Year

Based on the data set, it appears that the number of fake articles was highest during 2016, exceeding 1,000 every month

that year. Figure:[8] which displays the distribution of true or genuine articles, shows a low count until 2017, with only a few articles published during the earlier years of the data set. However, there was a sudden increase in the number of true articles in 2017, leading to a left-skewed distribution.

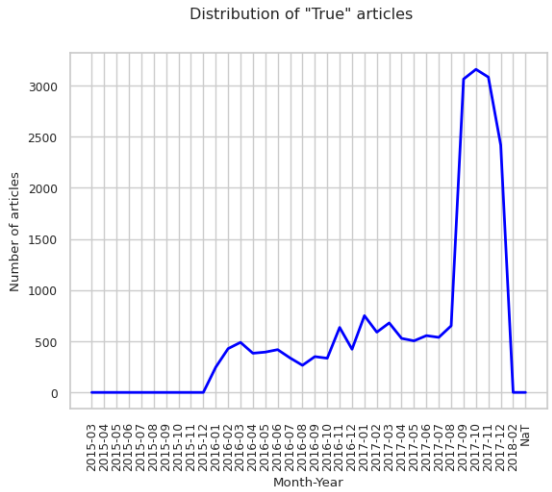

Figure 8. Distribution of True articles by month and year

Figure:[9], Figure:[6] show us the spread of a various number of articles over different categories such as politicsNews, world news, News, politics, Government News, and left news etc., we can see that the political category has the highest number of articles, exceeding 10,000. This may suggest that political news is a significant focus of the dataset, and the articles may cover various political events, issues, or topics. The high number of articles in the political category could also reflect the interest and engagement of readers or consumers of news with political news.

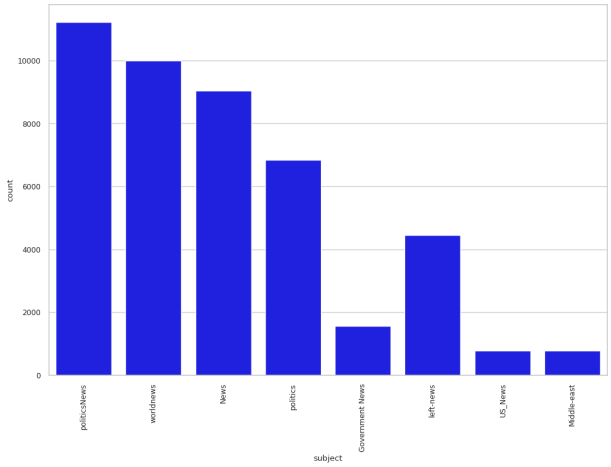

Figure 9. Distribution of True, Fake news by categories

The second and third highest categories, as mentioned, are world news and news, respectively. The world news category may contain articles that cover events, developments, or issues that have a global impact or interest, such as international conflicts, and economic trends. The news category, on the other hand, may encompass a broad range of topics, including political news, but also other types of news such as sports or entertainment.

The plots in Figure:[10], Figure:[11], shows the distribution of articles based on their class, either true or fake, over a period of time, similar to the previous line graph. These plots enable us to gain valuable insights into the spread of true and fake articles during specific situations.

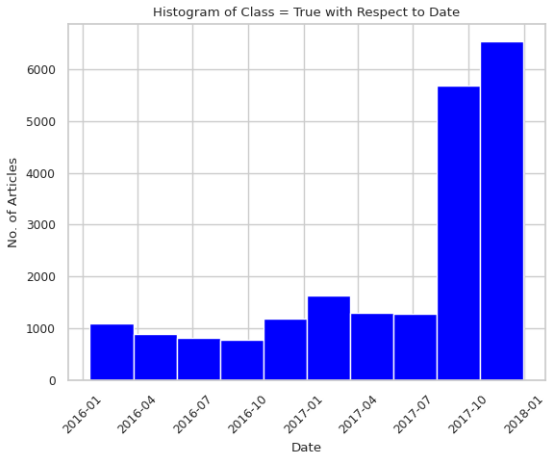

Figure 10. Distribution of True news by date

From Figure:[11], we can observe that the number of fake articles was higher from January 2016 to January 2017. The distribution of these fake articles during that period resembled a normal or Gaussian distribution. This suggests that there was a relatively consistent spread of fake articles during that time frame. The plots provide important information on the distribution of true and fake articles during various time periods, enabling us to identify patterns and trends. The higher prevalence of fake articles during a specific time frame may indicate a particular event or situation that triggered the spread of misinformation.
To understand which are the most frequent words appearing in the text to both the classes - ”True” and ”Fake”, we have created word clouds. Word clouds, represented in Figure:[12], Figure:[13] can provide some insight into the types of topics and language used during a particular time period.

It is evident that words like ”trump”, ”nation”, ”states”, ”president” etc are common in both the word clouds, we may infer that during the time of Trump and the presidential elections, a lot of fake news was in the air. This suggests that during Trump’s presidency and the presidential elections, there was a significant amount of fake news being circulated.

To understand if the author of fake news whose text disagrees with the title uses more number of words in the text

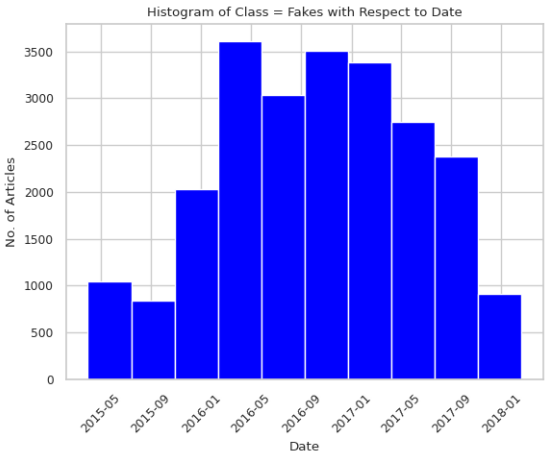

Figure 11. Distribution of fake news by date

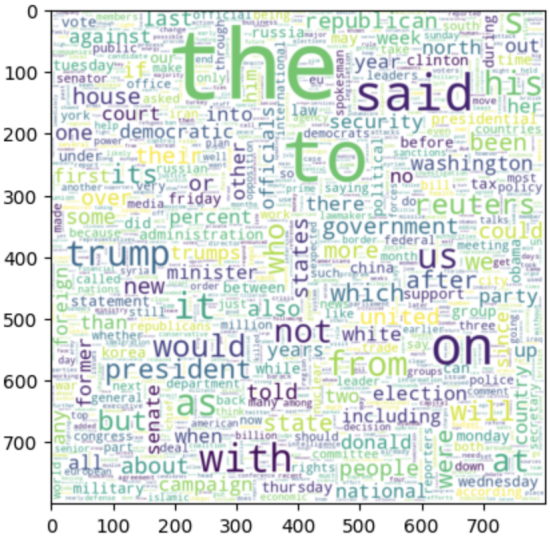

Figure 12. Word cloud for most frequent words appearing in True news

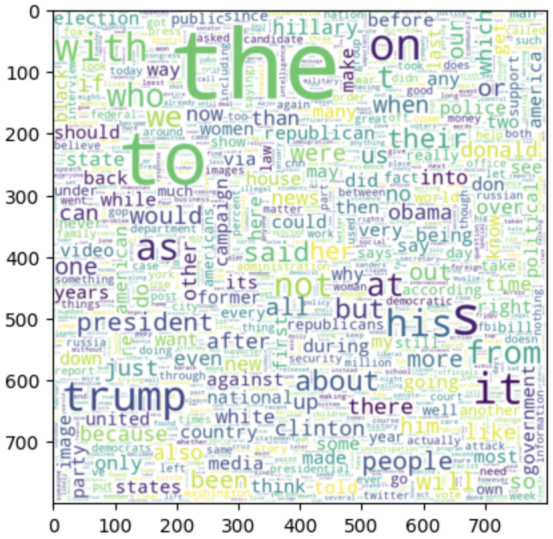

Figure 13. Word cloud for most frequent words appearing in Fake news

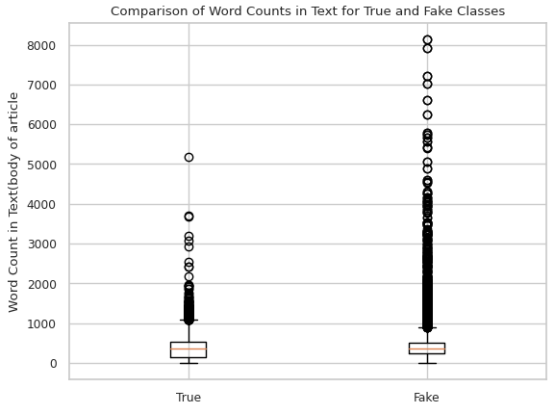

Figure 14. Comparison for word count in Text(body)

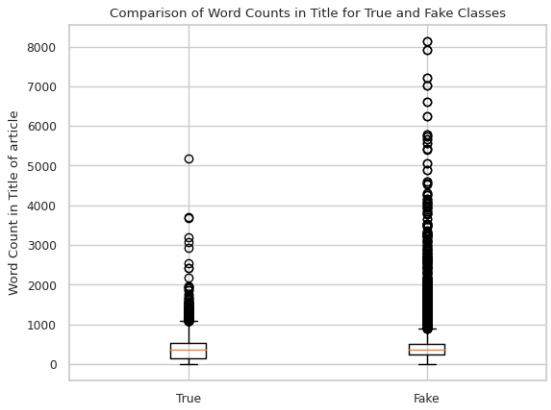

Figure 15. Comparison for word count in title

and title as compared to those where there is an agreement, we have calculated the average number of words in the text and the title.

We observed that from Figure:[14], Figure:[15] the average number of words used in the text and title when there is an agreement is at 384.75 and 9.95 respectively. The average number of words used in the text and title when there is a disagreement are 423.22 and 14.73 respectively. It is evident that the author tends to use more number of words in the text as well as title to prove his point or to keep the readers engaged when there is no relation between the title and the text. It is simpler when there is a flow between the title and the text. The scatter plot [16] reveals a positive correlation between the length of the title and the length of the text in the dataset. The observed upward trend of data points signifies that, on average, the length of the text increases with the length of the title. However, the wide distribution of points suggests that there are other factors influencing the length of the text beyond the title length. This plot provides a useful tool for researchers studying the relationship between the length of

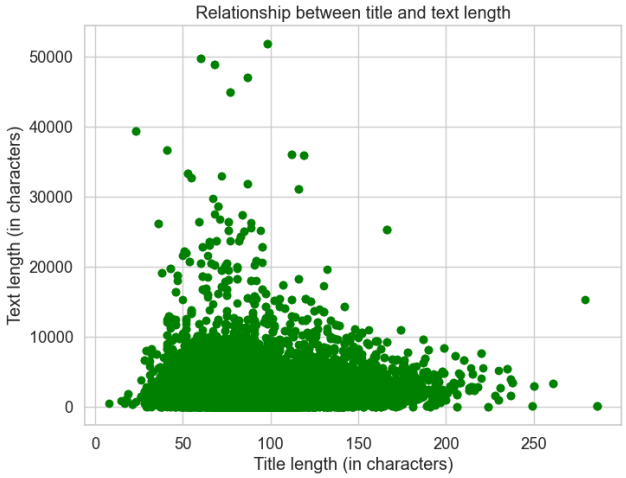

Figure 16. Relation between title and text length

the title and text length and may inform future investigations of the factors that impact text length.

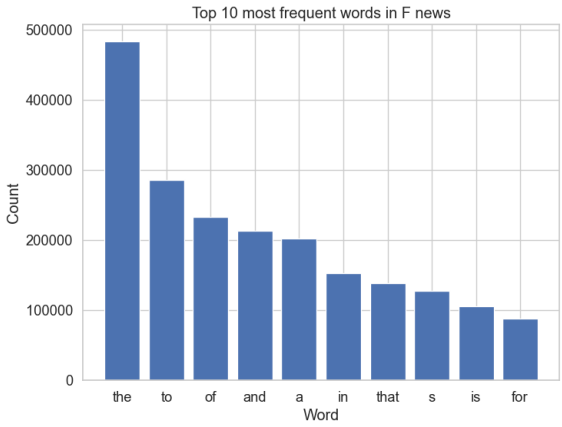

Figure 17. Top 10 most frequent words in Fake news

In this study, we utilized a frequency analysis technique to investigate the most frequently occurring words in two distinct types of news articles: fake and real news. The frequency analysis was conducted by first aggregating all news articles that belonged to the same category (i.e., fake or real news) and concatenating their text into a single corpus. From this corpus, the top 10 most frequently occurring words were then identified and plotted in a bar graph, with each bar representing a unique word and its corresponding frequency count.

From Figure:[17] and Figure:[18], we can observe that in the top 10 most frequently repeated words for both the true and fake data; Most of the words that are repeated are function words. Hence, we need to ignore such terms while further classifying the data to know the exact frequency of the nouns that are used in the text.

Aiming to investigate the relationship between several features extracted from news articles and the presence of fake

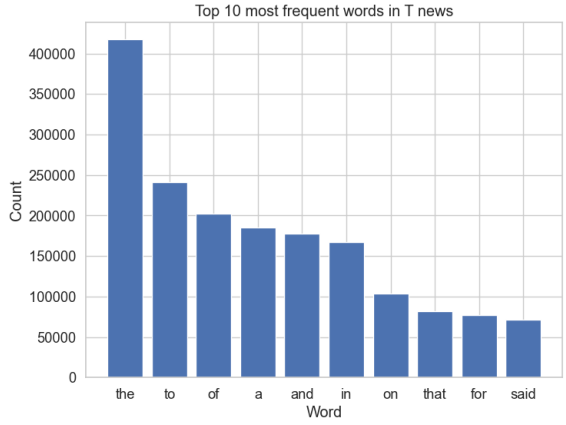

Figure 18. Top 10 most frequent words in True(reliable) news

news, we computed three additional features in the dataset, namely the length of the text and title of each news article, and the number of exclamation marks used in the text. From Figure:[19], our findings suggest that there is a weak positive correlation between the length of the text and the presence of fake news, while the correlation between the length of the title and the number of exclamation marks and the presence of fake news is negligible.

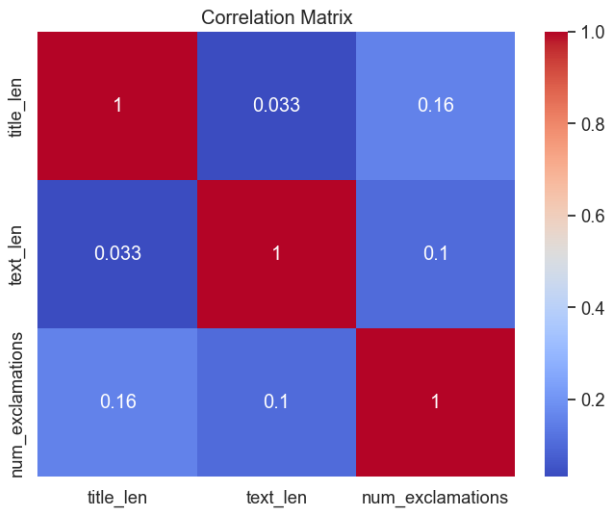

Figure 19. Correlation Matrix

These results can be informative for developing fake news detection models based on text mining techniques. For example, in the case of text length, our findings suggest that fake news may have a tendency to be longer than real news. This could be related to the fact that fake news often includes more details, descriptions, and claims that require more words to convey. Furthermore, our results suggest that the number of exclamation marks may not be a useful feature for detecting fake news, as the correlation with the presence of fake news

is negligible. Overall, our study provides insights into the relationship between certain features of news articles and the presence of fake news, which can be helpful for developing more effective fake news detection models.

Below is the Eda of dataset used for stance detection:

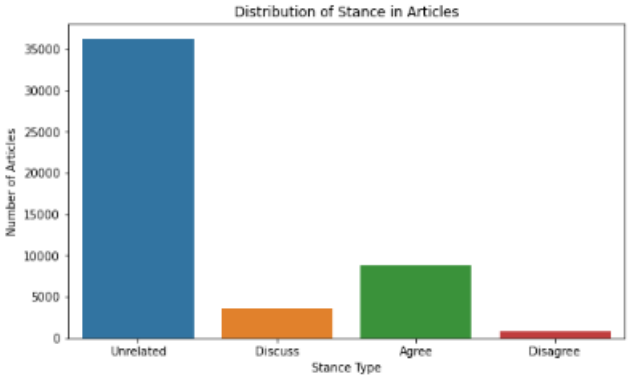

Figure 20. Distribution of stance

From figure[20], It can be noted that the distribution of the Target class in the dataset is imbalanced. Specifically, there are significantly more unrelated stances (36545) than discuss (8909), agree (3678), and disagree (840) stances. This means that the number of examples belonging to each category is not evenly distributed. The number of examples with the "unrelated" stance is much higher than the number of examples with the other stances.

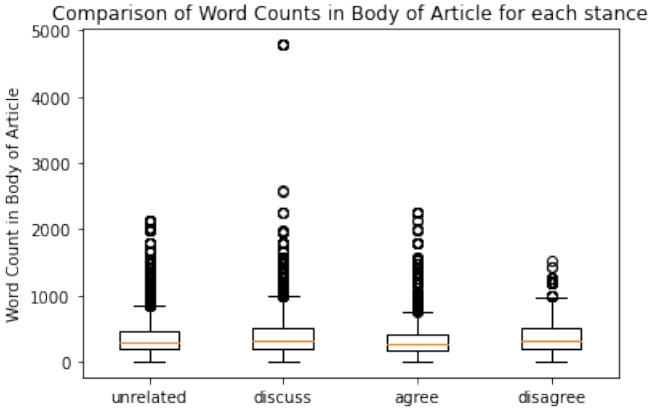

Figure 21. comparision of word counts in Body of article for each stance

From the Figure[21], we can observe the distribution of word counts in the Body of Article with respect to each, after removing unnecessary punctuations and lemmatization of words. From the Figure [22], we can observe the distribution of word counts in the Headline of Article with respect to each, after removing unnecessary punctuations and lemmatization of words.

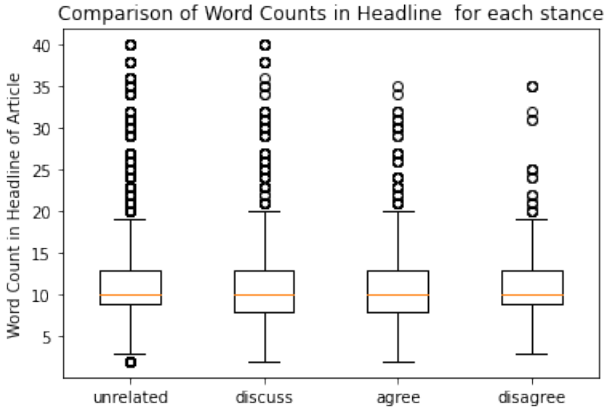

Figure 22. comparision of word counts in Headline of article for each stance

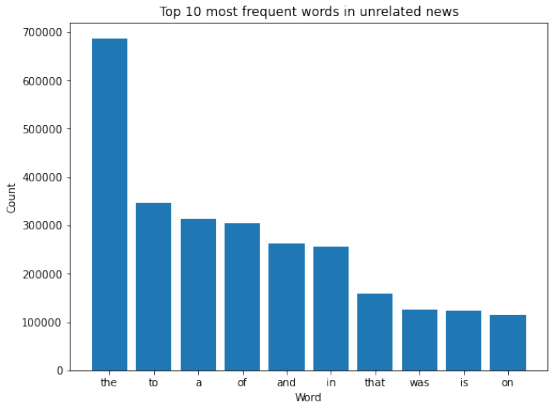

Figure 23. Top 10 most frequent in unrelated stance

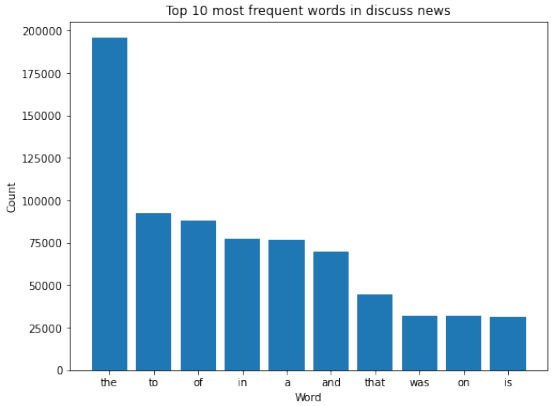

Figure 24. Top 10 most frequent in discuss stance

Figure[24] shows the Top 10 most frequent words that are appeared in Discuss stance, similarly Figure[23] shows the most frequent words that appeared with respect to unrelated stance, and Figure[25] shows the relationship between number of words in Headline and Article body length.

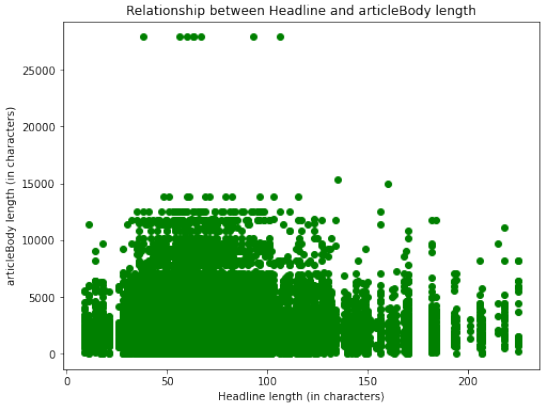

Figure 25. Relationship between Headline and article body length

2.3 Text Preprocessing

Basic data preprocessing includes:

Tokenization : The input string 's' is tokenized into a list of words using the wordtokenize() function from the NLTK library, which breaks the input string into individual words.

Lemmatization and lowercase: Each word token 't' in the tokenized list is then lemmatized and converted to lowercase. The lemmatized and lowercase word is appended to a new list, resulting in a list of normalized lemmas.

Cleaning the string: The input string s is processed using the re.findall() function from the re module, which searches for all occurrences of one or more alphanumeric characters in the input string.

Stop words removal : Removing a pre-defined set of stop words

Finding the overlapping words in headline and body text: First, clean both the headline and body text and convert the lemmatized headline and body tokens into sets, and the intersection() method is called to compute the common words between the sets using "set(cleanheadline).intersection(cleanbody)". Convert the lemmatized headline and body tokens into sets, and the union() method is called to compute the total words between the sets using " set(cleanheadline).union(cleanbody)". Further the length of the words in intersection and union are calculated. The total number of overlapping words are obtained by dividing the two values "len(set(cleanheadline). intersection(cleanbody)) / float(len(set(cleanheadline).union(cleanbody)))"

Generating n-grams: We have generated n-grams for the strings

Generating chargrams: We then generated character n-grams of length n

Creating handfeatures by counting the binary co-occurences features for each pair of headline and body in the input lists headlines and bodies.

After text preprocessing, we observed that the train data is highly biased so we brainstormed on how to handle this highly imbalanced dataset and came up with an idea of **Adaptive Synthetic Sampling(ADASYN)**. This approach works according to the density of the minority class instances. Generating new samples is inversely proportional to the density of the minority class samples. It generates more samples in the feature space region where minority class examples density is low or none and fewer samples in the high-density space. The before and after sampling results can be observed in the below figures[26] and [27]

```
unrelated    36545
discuss       8909
agree         3678
disagree       840
Name: Stance, dtype: int64
```

Figure 26. Before Sampling

```
agree        36852
discuss      36714
disagree     36694
unrelated    36545
Name: Stance, dtype: int64
```

Figure 27. After Sampling

3. Algorithm and Methodology

3.1 Hypothesis for stance detection

Null Hypothesis:The title and the body text aligns

Alternate Hypothesis: There is a difference between the title and the body text

3.2 Cross-validation

We have used this technique to evaluate the performance of a model and prevent overfitting. It involves partitioning the data set into multiple subsets or folds, where the model is trained on one subset and tested on another. The primary purpose of cross-validation is to ensure that the model generalizes well to new data and does not overfit the training data. It also helps mitigate selection bias by evaluating the model on a representative sample of the data. cross-validation is an essential technique to ensure the reliability of machine learning models and to prevent overfitting and selection bias.

3.3 Hyperparameter tuning

We used this technique to pick the optimal values for the hyperparameters of a machine learning algorithm to improve model performance. Hyperparameters are settings that are not learned from the data but are set by individuals prior to training the model. Examples of hyperparameters include learning rates, regularization strength, and a number of hidden layers. Hyperparameter tuning can help to improve the performance of a model by finding the optimal settings for these hyperparameters. Tuning can lead to a more robust model that generalizes well to new data, making it less prone to overfitting. A model with optimized hyperparameters is

more likely to achieve better accuracy, precision, and recall on the test dataset.

3.4 Algorithm for Stance Detection

```
Input: Headline and Text Body
Output: Stance {unrelated, discuss, agree, disagree}
Pseudocode:
TRANSFORM_HEADLINE_TO_VECTOR (data['headline'])
TRANSFORM_TEXT_TO_VECTOR (data['text'])
CONCATENATE_VECTORS (data_head_vec, data_body_vec)
COMPUTE_HAND_FEATURES (data['headline'], data['text'])
CONCATENATE_VECTORS (data_final,data_handF)
PREDICT_WITH_ENSEMBLE_MODEL (data_final)
CREATE_LABEL_MAPPING ()
GET_FIRST_PREDICTED_VALUE (ensemble_preds)
IF result_value EXISTS IN label_mapping. values () THEN
 reversed_mapping = REVERSE_LABEL_MAPPING (label_mapping)
 stance = GET_STANCE_FROM_REVERSED_MAPPING (reversed_mapping,
result_value)
ELSE
 stance = 'unknown'
END IF
RETURN stance
```

Figure 28. Stance Algorithm

The stance detection algorithm takes a headline and text body as input and transforms them into numerical vectors. These vectors are then combined and some hand-crafted features are computed based on the input. The resulting vectors and features are fed into a machine-learning model, which predicts the stance of the headline and text body.

The model outputs a numerical value, which is then mapped to a corresponding stance label using a label-mapping dictionary. The possible stance labels are unrelated, agree, disagree, or discuss. If the predicted value is not present in the label mapping, then the algorithm outputs 'unknown' as the stance.

To summarize, the algorithm analyzes two pieces of text and predicts whether they are related, in agreement, in disagreement, or open for discussion. It does this by transforming the input into numerical vectors, combining them with hand-crafted features, and feeding them into a machine-learning model.

Figure[29] Describes the outline of the Model that we have implemented. On taking the title and body of an article in the form of text input,we preprocessed the input by using techniques like vectorization, tokenization, normalization etc. We then passed this input to a user-defined function hand features that clean the string, count the binary occurrences of the words in the input with and without the stopword removal, count the grams in the input, and remove the overlapping features. We also calculated the Union of the common words found through their count of binary occurrences. After preprocessing this data, we then split the data into train and test

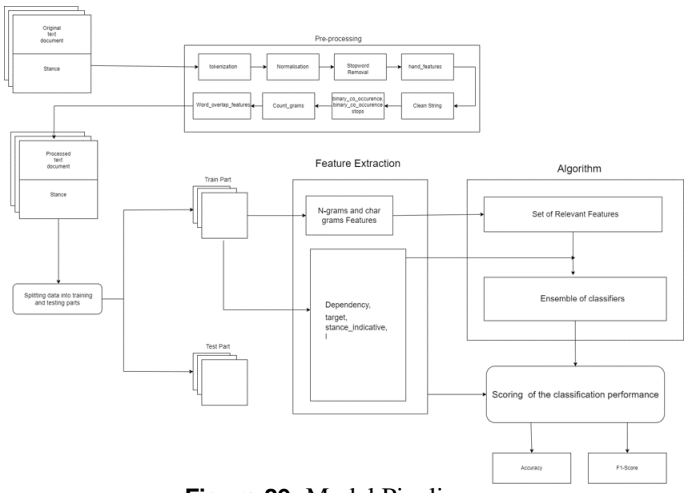

Figure 29. Model Pipeline

parts and calculated the n-grams and char grams of the words we got from the processed document, then we have run the trained data on various classifiers and classified them based on the target variable which is the stance indicative. We then took the ensemble of all these classifiers through voting and calculated the performances of this model based on the F1 score and accuracy.

4. Experiments and Results

We build various machine learning models like **Random Forest Classifier**,**Naive Bayes**, **Support Vector Machines**, **K-nearest neighbors**, and a deep learning model.

4.1 Random Forest Classifier

The param_dist dictionary contains a wide range of hyperparameter values for the RandomForestClassifier, including the number of estimators (n_estimators), maximum number of features to consider at each split (max_features), whether or not to use bootstrapping (bootstrap), class weights (class_weight), splitting criterion (criterion), maximum depth of the trees (max_depth), the minimum number of samples required to be a leaf node (min_samples_leaf), and minimum number of samples required to split an internal node (min_samples_split). The specific hyperparameter values and their ranges can be adjusted according to the requirements of the problem being solved. We fitted 10 folds for each of 10 candidates, totaling

```
random_search = RandomizedSearchCV(clf, param_distributions=param_dist,
                                   n_iter=n_iter_search,
                                   scoring='f1_micro',
                                   cv=10,
                                   n_jobs=-1,verbose=20)
```

Figure 30. Random Forest Cross Validation

100 fits.Figure[30] shows the result of the cross-validation.
1. The Randomizedserach CV randomly samples a combination of hyperparameter values from the specified hyperparameter distributions (param_distributions) for a given number of

```
RandomizedSearchCV(cv=10, error_score='raise-deprecating',
          estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
          fit_params=None, iid='warn', n_iter=10, n_jobs=-1,
          param_distributions={'n_estimators': [100, 200, 300, 500, 700, 1000], 'max_features': [20, 40, 60, 80,
100, 120, 140, 160, 187], 'bootstrap': [True, False], 'class_weight': [None, 'balanced'], 'criterion': ['entrop
y', 'gini'], 'max_depth': [None, 5, 10, 15, 20, 30, 50, 70], 'min_samples_leaf': [1, 2, 5, 10, 15, 20], 'min_sam
ples_split': [2, 5, 10, 15, 20]},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring='f1_micro', verbose=20)
```

Figure 31. Hyperparameter Tuning

iterations (n_iter).

2. Fits the estimator (clf), which is the RandomForestClassifier in this case, with the sampled hyperparameter values to the training data (train_final) and labels (train_labels). Performs cross-validation with the specified number of folds (cv) to evaluate the performance of the estimator with the sampled hyperparameter values. Cross-validation is a technique used to estimate the performance of a model on unseen data by splitting the training data into multiple folds and training the model on different folds while evaluating its performance on the remaining folds.

3. Computes the evaluation scores (based on the scoring metric specified by scoring) for each combination of hyperparameter values.

4. Keeps track of the best combination of hyperparameter values found during the search based on the evaluation scores.

Once all iterations are completed, the best hyperparameter values are stored in the best_params_ attribute of the RandomizedSearchCV object, and the best estimator (i.e., the estimator trained with the best hyperparameter values) is stored in the best_estimator_ attribute. We also checked the important

```
Model with rank: 1
Mean validation score: 0.89134 (std: 0.00654)
Parameters: {'n_estimators': 100, 'min_samples_split': 10, 'min_samples_leaf': 15, 'max_features': 60, 'max_depth':
30, 'criterion': 'entropy', 'class_weight': None, 'bootstrap': False}

Model with rank: 2
Mean validation score: 0.88898 (std: 0.00752)
Parameters: {'n_estimators': 100, 'min_samples_split': 15, 'min_samples_leaf': 5, 'max_features': 100, 'max_depth':
30, 'criterion': 'gini', 'class_weight': None, 'bootstrap': True}

Model with rank: 3
Mean validation score: 0.88890 (std: 0.01067)
Parameters: {'n_estimators': 500, 'min_samples_split': 20, 'min_samples_leaf': 1, 'max_features': 60, 'max_depth':
30, 'criterion': 'entropy', 'class_weight': None, 'bootstrap': False}

Model with rank: 4
Mean validation score: 0.88860 (std: 0.00838)
Parameters: {'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf': 5, 'max_features': 100, 'max_depth':
70, 'criterion': 'gini', 'class_weight': None, 'bootstrap': True}

Model with rank: 5
Mean validation score: 0.88834 (std: 0.00726)
Parameters: {'n_estimators': 1000, 'min_samples_split': 5, 'min_samples_leaf': 15, 'max_features': 60, 'max_depth':
15, 'criterion': 'gini', 'class_weight': None, 'bootstrap': False}
```

Figure 32. Report showing the best five Random Forest Models

features and the accompanying image shows importamce of each feature. We obtained an training accuracy of 94.77%

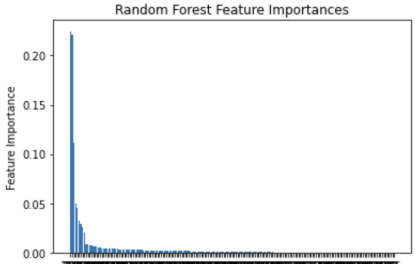

Figure 33. Feature Importance

and test accuracy of 86.23%. The following image shows the confusion matrix of the same.

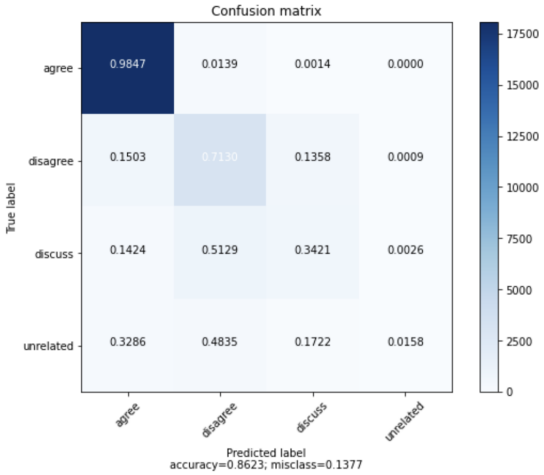

Figure 34. Confusion Matrix

4.2 Neural Network

We build a deep learning model. Model summary is shown in Figure[35] below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 187)               0
_________________________________________________________________
dense_1 (Dense)              (None, 200)               37600
_________________________________________________________________
dense_2 (Dense)              (None, 100)               20100
_________________________________________________________________
dense_3 (Dense)              (None, 40)                4040
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 164
=================================================================
Total params: 61,904
Trainable params: 61,904
Non-trainable params: 0
_________________________________________________________________
```

Figure 35. Model Summary

Epochs after fitting the model are displayed in Figure[36]

```
Epoch 1/5
49972/49972 [==============================] - 9s 186us/step - loss: 0.3147 - acc: 0.8827
Epoch 2/5
49972/49972 [==============================] - 8s 163us/step - loss: 0.2677 - acc: 0.9017
Epoch 3/5
49972/49972 [==============================] - 8s 162us/step - loss: 0.2543 - acc: 0.9052
Epoch 4/5
49972/49972 [==============================] - 8s 164us/step - loss: 0.2413 - acc: 0.9088
Epoch 5/5
49972/49972 [==============================] - 8s 163us/step - loss: 0.2293 - acc: 0.9139
```

Figure 36. Epochs

The accuracy of the deep learning model is 86.30%. Confusion matrix is shown in Figure [37].

4.3 Naive Bayes

We have used the training set to train the Multinomial Naive Bayes classifier with alpha =0.34. [Figure 38f]]

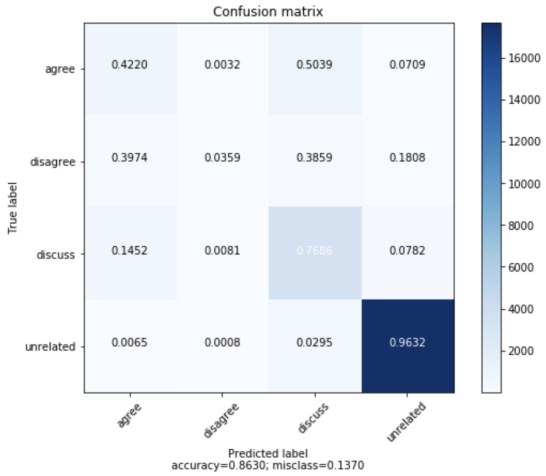

Figure 37. Confusion matrix

```
MultinomialNB(alpha=0.34, class_prior=None, fit_prior=True)
```

Figure 38. Multinomial Naive Bayes

Naive Bayes makes the assumption that the features used for classification are independent of each other, given the class label. This simplifies the computation of the probabilities. Further, we used the training data to estimate the parameters of the model - class prior probabilities and the conditional probability distributions for each feature given the class labels. We evaluated the performance of the trained Naive Bayes classifier on the testing set using F1-score.
The accuracy of the Naive Bayes Classifier is 82.56%.
The confusion matrix for Multinomial Naive Bayes Classifier is shown in Figure [39].

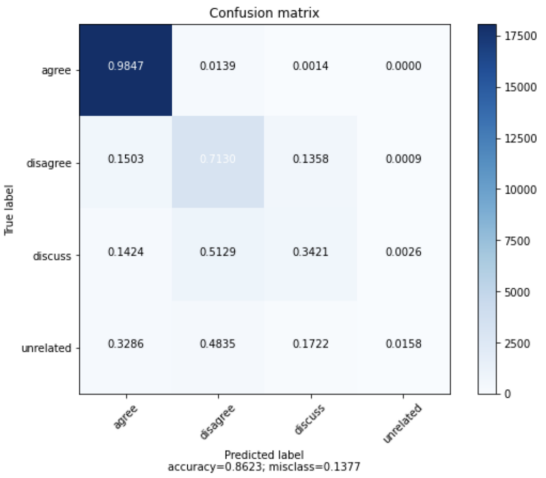

Figure 39. Confusion matrix

4.4 Support Vector Machines

To classify the correct stance SVM aims to find a hyperplane that best separates the data points of different stances in the feature space. Training set is used to train the SVM model. We experimented with two different SVM kernels polynomial, and radial basis function (RBF).
After training the SVM model, we evaluated the performance of the model on the test data set and used F1-score as the evaluation metric. We also tried to fine-tune the model by adjusting the hyperparameter C-regularization parameter and class weights to optimize the model performance.
The training accuracy was 87.79% and testing accuracy was 85.89%.
The confusion matrix is shown in Figure [40].

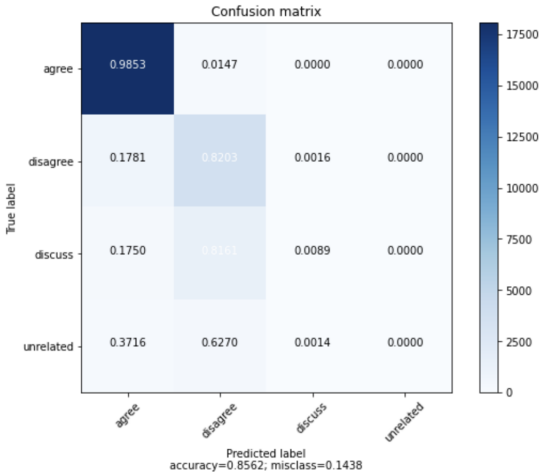

Figure 40. Confusion matrix

4.5 K-nearest neighbors

We trained a K-nearest neighbor model using the training set. During training, KNN computes distances between data points and store them so as to classify the stance correctly later on. We tuned the hyperparameters used in the KNN algorithm, like K -number of neighbors to consider and the distance metric used. This involved us experimenting with different values of K and different distance metrics to find the best combination for classifying the stance correctly.
We evaluated the trained KNN model on the validation dataset using F1 score as the evaluation metric for stance detection.
We predicted the stance on new, unseen test data to perform stance detection classification with an accuracy of 84.87%.
The confusion metric is shown below in Figure [41].
The following table Figure [42] compares the training and testing accuracy of all the models built.

4.6 Ensemble Model

The **idea** behind creating this ensemble model is to combine the predictions of multiple models created earlier, the resulting predictions can be more accurate and robust compared to using a single model. This is because the individual models in the ensemble may have different strengths and weaknesses, and by combining them, the ensemble can capitalize on the strengths of each individual model while mitigating their weaknesses.

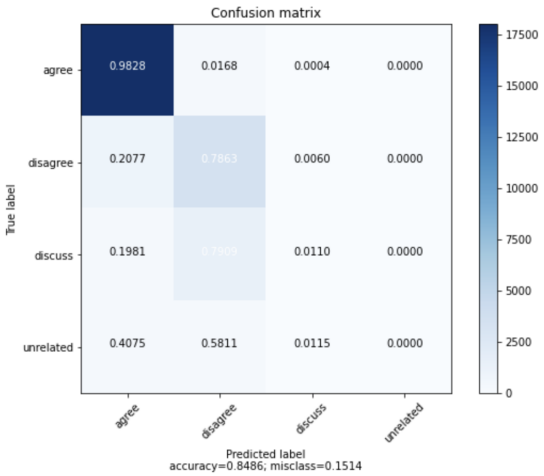

Figure 41. Confusion matrix

Model	Test Accuracy	Train Accuracy
Random Forest	86.23%	94.77%
Naive Bayes	86.24%	88.02%
Neural nets	86.59%	88.49%
SVM	85.89%	87.79%
KNN	84.87%	87.02%

Figure 42. Comparision of accuracies

We have used hard voting for classifying, the final prediction of the stance is therefore determined by a simple majority vote among the base models. We have used F1 score as the evaluation metric and used F1 micro to evaluate the same. In F1 micro, the precision, recall, and F1 score are calculated globally by considering all the classes as a single class. That is, the counts of true positives, false positives, and false negatives are aggregated across all classes, and a single F1 score is calculated based on these aggregated counts. F1 micro gives equal importance to all classes and treats them as a whole. Below is the flow diagram representation of our Ensemble model.

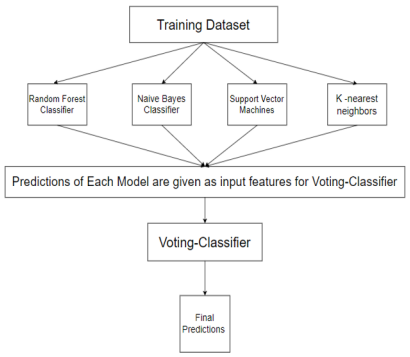

Figure 43. Ensemble model

```
Classifier(estimators=[('rf',
                        RandomForestClassifier(bootstrap=False,
                                               criterion='entropy',
                                               max_depth=30,
                                               max_features=60,
                                               min_samples_leaf=15,
                                               min_samples_split=10)),
                       ('svm', SVC()), ('nb', MultinomialNB(alpha=0.34)),
                       ('knn', KNeighborsClassifier(n_neighbors=100))])
Ensemble Model Accuracy (F1 micro): 0.8557431235981584
```

Figure 44. Ensemble model F1 macro Accuracy

```
Pearson's corr coef b/n ensemble and target:  0.7217953088138392
Pearson's corr coef b/w rf and target:  0.7381284642609152
Pearson's corr coef b/w nb and target:  0.6515083724522523
Pearson's corr coef b/w knn and target:  0.5848703702082666
Pearson's corr coef b/w svm and target:  0.7246966238018951
```

Figure 45. Pearson corr between various models and Target

	precision	recall	f1-score	support
0	0.61	0.97	0.75	7044
1	0.91	0.31	0.46	6363
accuracy			0.66	13407
macro avg	0.76	0.64	0.60	13407
weighted avg	0.75	0.66	0.61	13407

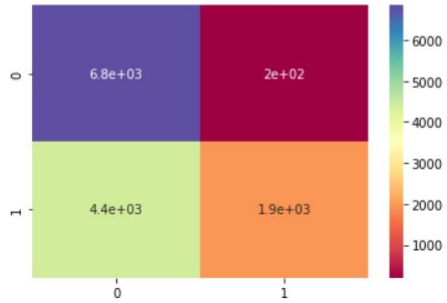

Figure 46. Metrics of KNN

	precision	recall	f1-score	support
0	0.70	0.98	0.81	7044
1	0.96	0.52	0.68	6363
accuracy			0.76	13407
macro avg	0.83	0.75	0.75	13407
weighted avg	0.82	0.76	0.75	13407

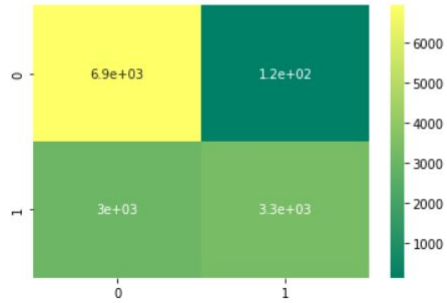

Figure 47. Metrics of Adaboost with RF

With our Ensemble model we have achieved an Accuracy (F1 Micro) of **85.57%** and we can observe the parameters of

	precision	recall	f1-score	support
0	0.56	0.98	0.72	7044
1	0.90	0.16	0.27	6363
accuracy			0.59	13407
macro avg	0.73	0.57	0.50	13407
weighted avg	0.72	0.59	0.51	13407

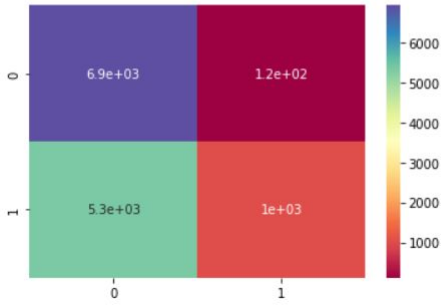

Figure 48. Metrics of XGBoost

	precision	recall	f1-score	support
0	0.61	0.98	0.75	7044
1	0.93	0.30	0.46	6363
accuracy			0.66	13407
macro avg	0.77	0.64	0.61	13407
weighted avg	0.76	0.66	0.61	13407

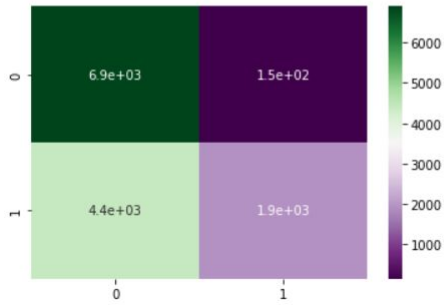

Figure 49. Metrics of Randomn Forest

our model and correlations of our models with our target. It can be noted that in most instances of running the program, the stacked predictions outperform any singular model. After following almost the same pipeline for true/fake dataset we achieved the below Evaluation metrics for KNN, AdaBoost with random forest, XGBOost with random forest and Ensemble of Naive Bayes as shown in the below figures. Figure 51,52,53 are screenshots of the prediction of stances of the articles from popular websites like "Nature.com", "New York Times".

The model has been thoroughly tested with different test cases for both true/fake classification and stance detection tasks, and in most cases, it gave accurate results. This implies that the model has been trained well and is able to handle real-time data effectively.

Testing a machine learning model with various test cases is an essential part of model development as it helps to identify

	precision	recall	f1-score	support
0	0.99	0.90	0.95	7044
1	0.90	0.99	0.95	6363
accuracy			0.95	13407
macro avg	0.95	0.95	0.95	13407
weighted avg	0.95	0.95	0.95	13407

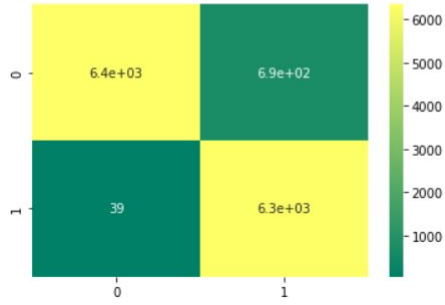

Figure 50. Metrics of Ensemble of Naive Bayes

```
stance_detection('The Indian Premier League (IPL)','The Indian Premier League (IPL) (also known as the TATA IPL
headline: The Indian Premier League (IPL)
text: The Indian Premier League (IPL) (also known as the TATA IPL for sponsorship reasons) is a mens Twenty20 (T20)
cricket league held annually in India and contested by ten city-based franchise teams.[3][4] The league was founded
by the Board of Control for Cricket in India in 2007. The competition is usually held in the summer months (between
March and May) every year and has an exclusive window in the ICC Future Tours Programme, meaning that less internat
ional cricket take place during the IPL seasons.[5]
1it [00:00, 268.16it/s]
'agree'
```

Figure 51. Example 1 of Prediction of stance

```
stance_detection('IBM Corp expected to cut more than 110,000 jobs in massive overhaul, report says','A female Ku
headline: IBM Corp expected to cut more than 110,000 jobs in massive overhaul, report says
text: A female Kurdish fighter, who was viewed as an iconic face of the Kurdish resistance movement, has been repor
tedly beheaded by ISIS.

The woman, known by the nickname â??Rehanaâ?? became a face of the conflict after a picture of her holding a rifle
and making a V sign was posted on Twitter, receiving over 5,000 retweets.

She fought for the Kurdish Womenâ??s Defense Unit and claimed to have killed 100 ISIS militants.

It is now believed that Rehana may have been killed by IS militants after a picture circulated on Twitter, purporti
ng to show her head being held aloft by a member of the jihadist organisation.

Her death has not been confirmed, however, by the Womenâ??s Defense Unit.

She had been fighting to defend Kobane, a Syrian border town which has been under assault by ISIS for more than a m
onth.

More than 10,000 women have helped to defend the town since the establishment of all female combat units in April.

They now reportedly play a major role in fighting against the IS.

Another female Kurdish fighter, who goes by the pseudonym of Afshin Kobani, said that the advancement of ISIS in Sy
ria had forced to her to join the fight in Kobane.

â??I lost many friends to this, and I decided there was a need to join up,â??

â??This is our land â?? our own â?? and if we donâ??t do it, who else will?

â??We are just the same as men; thereâ??s no difference,â?? she said. â??We can do any type of job, including armed
mobilisation.â??
1it [00:00, 134.33it/s]
'unrelated'
```

Figure 52. Example 2 of Prediction of stance

```
stance_detection('Low-Energy Intense Pulsed Light for Hair Removal at Home','Low-energy intense pulsed light for
headline: Low-Energy Intense Pulsed Light for Hair Removal at Home
text: Low-energy intense pulsed light for hair removal at home was evaluated in this clinical trial. Twenty-two fem
ale patients were enrolled into an institutional review board-approved clinical trial. Patients received six biweek
ly treatments with the device, and clinical results with hair counts and pictures were performed at four weeks and
three months following the last treatment. Ninety-five percent of the patients noted hair count reduction at the en
d of this clinical trial. Overall hair reduction was 78 percent at the one-month follow up and 72 percent at the th
ree-month follow up. No serious adverse events were noted. This clinical trial confirmed the safety and efficacy of
this device for hair removal at home.
1it [00:00, 228.62it/s]
'agree'
```

Figure 53. Example 3 of Prediction of stance

potential issues and improve the model's performance. By testing the model with different cases, developers can understand how the model behaves with different data inputs and can fine-tune the model to handle edge cases and corner cases.

In the case of true/fake classification and stance detection, accurate results are crucial to prevent the spread of misinformation and to maintain the integrity of the news industry. Therefore, the fact that the model was able to perform accu-

rately in most cases is a promising sign that it can be deployed in real-world scenarios to help combat the spread of fake news.

5. Deployment and Maintenance

Our stance detection project is currently live on a website that has an attractive and easy-to-navigate user interface. The machine learning API endpoint for this project is hosted locally at "http://localhost:8080".and clinet and server links are
Client: `https://github.com/avinash-rdy2903/dm-client`
Server: `https://github.com/avinash-rdy2903/stance-detection-serve`
Figure[54] shows our deployed website.We have implemented two endpoints for the stance detection project:

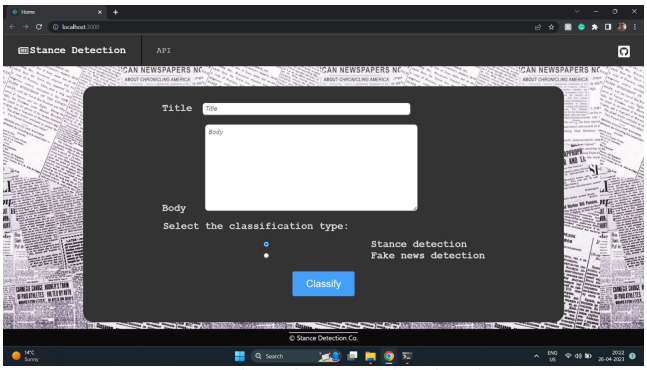

Figure 54. Metrics of Ensemble of Naive Bayes

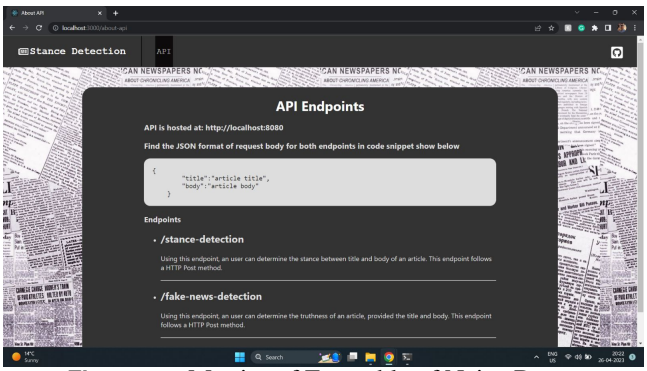

Figure 55. Metrics of Ensemble of Naive Bayes

/stance-detection: This endpoint uses a pre-trained model to detect the stance of a given text towards a target entity. To use this endpoint, the client needs to send an HTTP POST request to the endpoint with a request body containing the text and the target entity. The endpoint returns a JSON response indicating the stance of the text towards the target entity.

/fake-news-detection: This endpoint uses a pre-trained model to detect whether a news article is real or fake based on its title and body. To use this endpoint, the client needs to send an HTTP POST request to the endpoint with a request body containing the title and body of the article. The endpoint returns a JSON response indicating whether the article is real or fake.

Both of these endpoints follow the HTTP POST request method, and the client needs to send a request body with the required input for the classifiers. The response is in the form of JSON format, which makes it easy to parse and integrate with other applications. The stance detection project is designed to be scalable, and we are continuously working on improving its performance and accuracy.

6. Summary and Conclusions

In this project, we have implemented of various machine-learning algorithms for the classification of stances in a given text. Random Forest, Neural Network, Naive Bayes, Support Vector Machines, and K-Nearest Neighbor algorithms and Ensemble method were used in this classification task. For Random Forest, Randomized Search CV was used to find the best hyperparameters and important features were extracted. The Multinomial Naive Bayes classifier was trained on the training set with alpha=0.34, and the Support Vector Machines used both polynomial and radial basis function kernels. The Random Forest algorithm performed the best with a training accuracy of 94.77% and a testing accuracy of 86.23%.

In conclusion, the proposed plan aims to improve the performance of the model for text classification tasks by incorporating state-of-the-art techniques such as contextualized pre-training. Additionally, the plan outlines strategies to optimize the server-side prediction process by leveraging batch operations, generating API keys to monetize the classification service, and automating the process of extracting the title and body from an article link for efficient classification. These steps are expected to enhance the overall usability, efficiency, and effectiveness of the text classification model, making it more competitive in the market.

Acknowledgments

1. We would like to acknowledge the support received from IU in providing us with valuable computing resources such as BIGRED to make our computing faster and more reliable in less time.

2. We would like to thank the dataset creators for providing the labeled dataset that we used to train and evaluate our model. This dataset was critical to the success of our research and allowed us to test our model's performance in a controlled environment.

3. We would like to thank our team members for their hard work and dedication in developing our machine-learning model. Their contributions were essential to the success of our project, and we are grateful for their support and collaboration.

4. We would like to acknowledge the contributions of the wider machine-learning community, whose work has inspired and informed our research. The wealth of knowledge and resources available in this field has been critical to our success and has allowed us to build upon the work of others to achieve new breakthroughs in Stance detection.

References

1. https://www.sciencedirect.com/science/article/pii/S0306457322001728#b45
2. https://ieeexplore.ieee.org/document/9552890?denied=
3. https://medium.com/@skillcate/detecting-fake-news-with-a-bert-model-9c666e3cdd9b
4. https://www.sciencedirect.com/science/article/pii/S187705092101797X
5. https://arxiv.org/pdf/2006.03644.pdf
6. https://www.sciencedirect.com/science/article/abs/pii/S0306457321000960?via%3Dihub