EGYPTATION E-learning University

El-fayoum center

Faculty of Information Technology

FAKE NEWS DETECTION

**Graduation Project**

Made by:

Ahmed Ashraf Fawaz Mahmoud ID: 20-00052

Salma Nasr Said Mahlhal ID: 20-01837

Eman Mohamed Nour Eldin ID: 20-00119

Nahla Ahmed Said Ibrahim ID: 20-01919

Mohab Mohamed Abdel-Azeem ID: 20-00202

Adnan Mohamed ID: 20-02033

Mohamed Mohsen Rashad ID: 20-01870

Supervised by:

Dr. Mohmmed Ashour

Eng. Radwa Sayed

Table of Contents

|  |  |
| --- | --- |
| Content | 1 |
| List of figures & tables | 2 |
| Chapter 1: Introduction | 5 |
| 1.1: What does Fake News mean? | 6 |
| 1.2: Project Problem Statement | 6 |
| 1.3: Project Motivation | 7 |
| 1.4: Project Aim and Objectives | 7 |
| 1.5: Project Requirements | 8 |
| 1.6: Project Expected Output | 8 |
| 1.7: Project Schedule | 9 |
| Chapter 2: Literature Review | 10 |
| 2.1: Introduction | 11 |
| 2.2: Existing Systems | 11 |
| 2.3: Solution Approach | 14 |
| Chapter 3: System Analysis | 18 |
| 3.1: Introduction | 19 |
| 3.2: Data Flow Diagrams | 20 |
| 3.3: UML Use Case Diagram | 23 |
| 3.3.1: UML Use Case scenario | 24 |
| 3.4: UML Sequence Diagram | 27 |
| 3.5: Class Diagram | 28 |
| 3.6: System Requirements | 29 |
| Chapter 4: The Dataset | 32 |
| 4.0 Introduction | 33 |
| 4.1: Dataset Definition | 33 |
| 4.2: Dataset Properties | 34 |
| Chapter 5: Proposed Model | 35 |
| 5.1: System Architecture | 36 |
| 5.2: Text Preprocessing | 38 |
| 5.3: Building Model | 43 |
| 5.4: News Classification | 45 |
| Chapter 6: Experimental results | 48 |
| 6.1: Confusion Matrix | 49 |
| 6.2: Classification Report | 51 |
| 6.3: RandomSearchCV and XGBoost | 51 |
| 6.4: Comparison between the proposed model and other models | 52 |
| Chapter 7: Conclusion and Future Work | 53 |
| 7.1: Conclusion | 54 |
| 7.2: Future work | 54 |
| Chapter 8: References | 55 |

Table of Figures

|  |  |
| --- | --- |
| **Figure name** | Number |
| Figure 1.1 : Project Schedule with tasks & participants | 9 |
| Figure 3.2.1 : System DFD Level 0 | 19 |
| Figure 3.2.2 : System DFD Level 1 | 20 |
| Figure 3.2.3 : System DFD Level 2 | 21 |
| Figure 3.2.4: System DFD Level 2 | 22 |
| Figure 3.3 : Use Case Diagram | 23 |
| Figure 3.3.1 : Use Case Scenario (1) | 24 |
| Figure 3.3.2 : Use Case Scenario (2) | 25 |
| Figure 3.3.3 : Use Case Scenario (3) | 26 |
| Figure 3.4 : Sequence Diagram | 27 |
| Figure 3.5 : Class Diagram | 28 |
| Figure 4 : Dataset | 34 |
| Figure 5.1 : System Architecture | 36 |
| Figure 5.4 : Front-End | 47 |
| Figure 6.1: Confusion Matrix | 50 |
| Figure 6.2: Classification Report | 51 |
| Figure 6.3: RandomSearchCV and XGBoost | 51 |
| Figure 6.4: Comparison between the proposed model and other models: | 52 |

Chapter 1

Introduction

* 1. **what does Fake News mean?**

Fake news or hoax news is false or misleading information presented as news. Fake news often has the aim of damaging the reputation of a person or entity or making money through advertising revenue.

* 1. **Project Problem Statement:**

In our modern era where the internet is ubiquitous, everyone relies on various online resources for news. Along with the increase in the use of social media platforms like Facebook, Twitter, etc. news spread rapidly among millions of users within a very short span of time. The spread of fake news has far-reaching consequences like the creation of biased opinions to swaying election outcomes for the benefit of certain candidates. Moreover, spammers use appealing news headlines to generate revenue using advertisements via clickbait. We aim to perform binary classification of various news articles available online with the help of concepts pertaining to Artificial Intelligence, Natural Language Processing and Machine Learning.

# Project Motivation:

Our motivation is simple, clear, and helpful to the world, which is detecting fake news. Because fake news could cause a distribution in a whole society, could affect a person or group of people with bad reputation, could cause hate to people or place by just telling and spreading fake news about them such as Palestine and people of Gaza, could also spread fake procedures to cure or protection from new disease that cause death to human being. Many examples are on the way and their impact on the subjects that are targeted is horrible and negative. And here comes our turn trying our best to face all this fake news specially when this kind of topic is now notice of interest because of all this spreading fake news.

# Project Aims and Objective:

The aims of fake news detection are centered around identifying, mitigating, and preventing the spread of false or misleading information, raising awareness, and enhancing fact checking process.

# Project Requirements

The system has special both software and hardware requirements:

## Hardware Requirements:

A personal device (PC or Laptop).

A computer with high “CPU” for model training.

## Software Requirements:

Web browser

Digital News from any platform and must be text

# Project Expected Output:

The expected output of the work outlined is to detect fake news, can be beneficial for anyone using our website to determine the news is fake or not through our model.

# Project Schedule

******

Figure 1.1: Project Schedule with tasks & participants

Chapter 2

Literature Review

# Introduction:

Now we will discuss some of what researchers and workers specified in this field have reached, the algorithms they used, the percentage of their work, and our solution or point of view in such a way that we can have a better understanding, higher performance, and percentage of our model.

# Existing System

Mykhailo Granik et. al. in their paper shows a simple approach for fake news detection using naive Bayes classifier. This approach was implemented as a software system and tested against a data set of Facebook news posts. They were collected from three large Facebook pages, each from the right and from the left, as well as three large mainstream political news pages (Politico, CNN, ABC News). They achieved classification accuracy of approximately 74%. Classification accuracy for fake news is slightly worse. This may be caused by the skewness of the dataset: only 4.9% of it is fake news. Himank Gupta et. All. gave a framework based on different machine learning approach that deals with various problems including accuracy shortage, time lag (BotMaker) and high processing time to handle thousands of tweets in 1 sec. Firstly, they have collected 400,000 tweets from HSpam14 dataset. Then they further characterize the 150,000 spam tweets and 250,000 non- spam tweets. They also derived some lightweight features along with the Top-30 words that are providing the highest information gain from Bag-of Words model. 4. They were able to achieve an accuracy of 91.65% and surpassed the existing solution by approximately 18%. Marco L. Della Vedova et. al. first proposed a novel ML fake news detection method which, by combining news content and social context features, outperforms existing methods in the literature, increasing its accuracy up to 78.8%. Second, they implemented their method within a Facebook Messenger Chabot and validate it with a real-world application, obtaining a fake news detection accuracy of 81.7%. Their goal was to classify a news item as reliable or fake; they first described the datasets they used for their test, then presented the content-based approach they implemented and the method they proposed to combine it with a social-based approach available in literature. The resulting dataset is composed of 15,500 posts, coming from 32 pages (14 conspiracy pages, 18 scientific pages), with more than International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Published by, [www.ijert.org](http://www.ijert.org/) NTASU - 2020 Conference Proceedings Volume 9, Issue 3 Special Issue - 2021 510 2, 300, 00 likes

by 900,000+ users. 8,923 (57.6%) posts are hoaxes and 6,577 (42.4%) are non-hoaxes. Cody Buntain et. al. develops a method for automating fake news detection on Twitter by learning to predict accuracy assessments in two credibility focused Twitter datasets: CREDBANK, a crowd sourced dataset of accuracy assessments for events in Twitter, and PHEME, a dataset of potential rumors in Twitter and journalistic assessments of their accuracies. They apply this method to Twitter content sourced from BuzzFeed’s fake news dataset. A feature analysis identifies features that are most predictive for crowd sourced and journalistic accuracy assessments, results of which are consistent with prior work. They rely on identifying highly retweeted threads of conversation and use the features of these threads to classify stories, limiting this work ‘s applicability only to the set of popular tweets. Since most]

tweets are rarely retweeted; this method therefore is only usable on a minority of Twitter conversation threads. In his paper, Shivam B. Parikh et. all. aims to present an insight of characterization of news story in the modern diaspora combined with the differential content types of news story and its impact on readers. Subsequently, we dive into existing fake news detection approaches that are heavily based on text- based analysis and describe popular fake news datasets. We conclude the paper by identifying 4 key open research challenges that can guide future research. It is a theoretical Approach which gives Illustrations of fake news detection by analyzing the psychological factors.

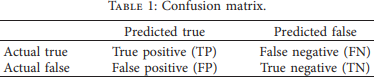
# solution Approach:

Our Contributions. In the current fake news corpus, there have been multiple instances where both supervised and unsupervised learning algorithms are used to classify text. However, most of the literature focuses on specific datasets or domains, most prominently the politics domain. Therefore, the algorithm trained works 2 Complexity best on a particular type of article’s domain and does not achieve optimal results when exposed to articles from other domains. Since articles from different domains have a unique textual structure, it is difficult to train a generic algorithm that works best on all news domains. We propose a solution to, as the learning models have the tendency to reduce error rate by using techniques such as bagging and boosting to facilitate the training of different machine learning algorithms in an effective and efficient manner. We also conducted extensive experiments on 4 real world publicly available datasets, results validate the improved performance of our proposed technique using the 4 commonly used performance metrics (namely, accuracy, precision, recall, and F-1 score).

the fake news detection problem using the machine learning ensemble approach. Our study explores different textual properties that could be used to distinguish fake

contents from real. By using those properties, we train a combination of different machine learning algorithms using various ensemble methods that are not thoroughly explored in the current literature. Ensemble learners have proven to be useful in a wide variety of applications

To evaluate the performance of algorithms, we used different metrics. Most of them are based on the confusion matrix. Confusion matrix is a tabular representation of a classification model performance on the test set, which consists of four parameters: true positive, false positive, true negative, and false negative (see Table 1).



* + 1. Accuracy is often the most used metric representing the percentage of correctly predicted observations, either true or false. To calculate the accuracy of a model performance, the following equation can be used:

[1.1](#_bookmark0)

In most cases, high accuracy value represents a good model, but because we are training a classification model in our case, an article that was predicted as true while it was actually false (false positive) can have negative consequences; similarly, if an article was predicted as false while it contained factual data, this can create trust issues.) Therefore, we have used three other metrics that consider the incorrectly classified observation, i.e., precision, recall, and F1-score.

* + 1. Recall represents the total number of positive classifications out of true class. In our case, it represents the number of articles predicted as true out of the total number of true articles.

A black and white math equation  Description automatically generated

* + 1. Precision score represents the ratio of true positives to all events predicted as true. In our case, precision shows the number of articles that are marked as true out of all the positively predicted (true) articles:



* + 1. F1-score represents the trade-off between precision and recall. It calculates the harmonic meaning between each of the two. ) It takes both the false positive and the false negative observations into account. F1-score can be calculated using the following formula:

A black text on a white background  Description automatically generated

Chapter 3

System Analysis

# Introduction:

This chapter represents System USE-CASE Scenario and USE-CASE Diagram which is a list of actions or events steps typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system to achieve a goal, Sequence Diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur, System Sequence Diagram which is a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events, Class Diagram which is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects, Context Diagram which presents the sub-systems of our system and its data flow processing, and system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

Fake News System

Real News

Fake News

News

Figure 3.2.1: System DFD Level 0

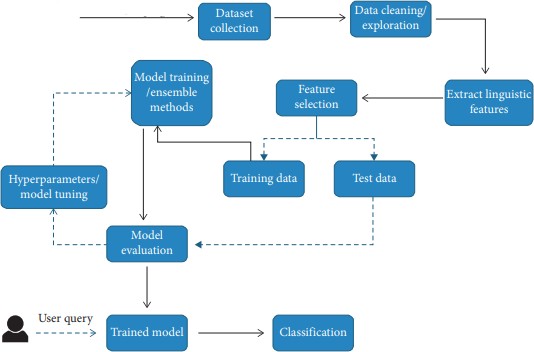


Figure 3.2.2: System DFD Level 1

Handline the missing values Scaling features

Check for imbalanced class

Feature engineering

Data Cleaning

Prepare data for XGBOOST

Explore categorical variable

Checking

duplicates

Figure 3.2.3: System DFD Level 2

Feature Extraction

Part of speech tagging

Bag of words

Named entity recognition

Sentiment analysis score

N-grams

Word Embaeddings

TF-IDF

Figure 3.2.4: System DFD Level 2

A diagram of a system

Description automatically generated

Figure 3.3: Use Case

|  |  |
| --- | --- |
| **Use case name** | **Data Cleaning** |
| **Actor(s)** | **Model** |
| **Description** | **The model receives the dataset from the admin and then begin in the procedures of cleaning such as handling the missing**  **values, checking for duplicates.** |
| **Typical of Events** |  |
| **Alternative** | **Step2: the data set uploaded is already cleaned and no need to enter the cleaning process** |
| **Precondition** | **Admin upload dataset need to be processed** |
| **Post condition** | **Admin will get the dataset is processed** |
| **Non- Functional requirement** | **All Non-Functional requirements: e.g., safety, reliability, and performance.** |

Figure 3.3.1: Use Case scenario 1

|  |  |
| --- | --- |
| **Use case name** | **Data Entry** |
| **Actor(s)** | **User** |
| **Description** | **The user uploads a text from an article in the determined place that the model will take on the process from.** |
| **Typical of Events** |  |
| **Alternative** | **Step2: if the text of the news entered in different**  **language, the model won’t start processing the news.** |
| **Precondition** | **User has selected a specified news.** |
| **Post condition** | **User receives the result of the prediction.** |
| **Non- Functional**  **requirement** | **All Non-Functional requirements: e.g., safety, reliability, and performance.** |

Figure 3.3.2: Use Case Scenario2

|  |  |
| --- | --- |
| **Use case name** | **Classification** |
| **Actor(s)** | **User & Actor** |
| **Description** | **In this process, which is considered last step in the system, the system brings output of the detection of the news either**  **it fake or not for the user.** |
| **Typical of Events** |  |
| **Alternative** | **Step 2: no alternative.** |
| **Precondition** | **User has selected a specified news.** |
| **Post condition** | **User receives the result of the prediction.** |
| **Non-**  **Functional requirement** | **All Non-Functional requirements: e.g., safety, reliability, and performance.** |

Figure 3.3.3: Use Case Scenario3

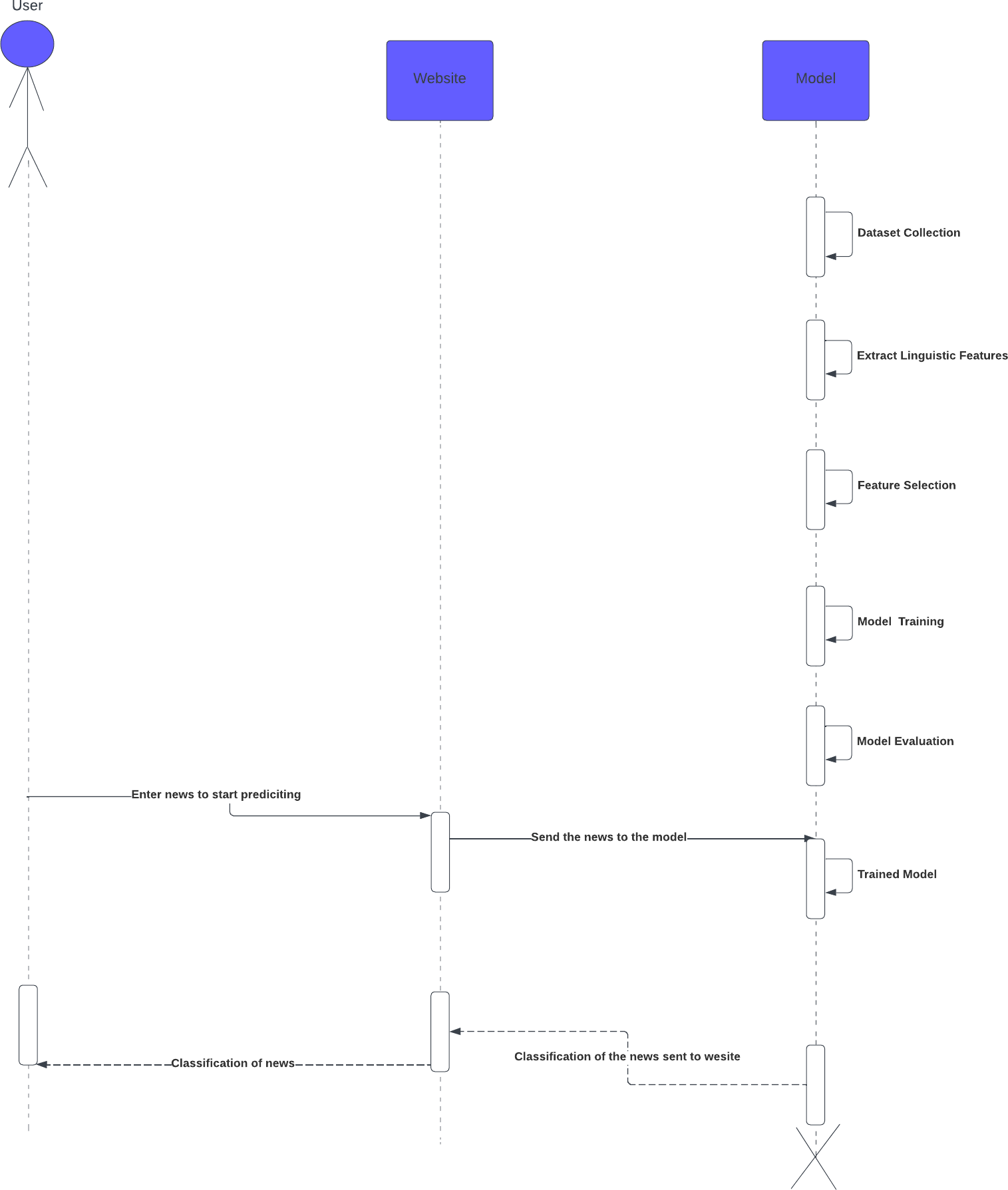


Figure 3.4: Sequence Diagram

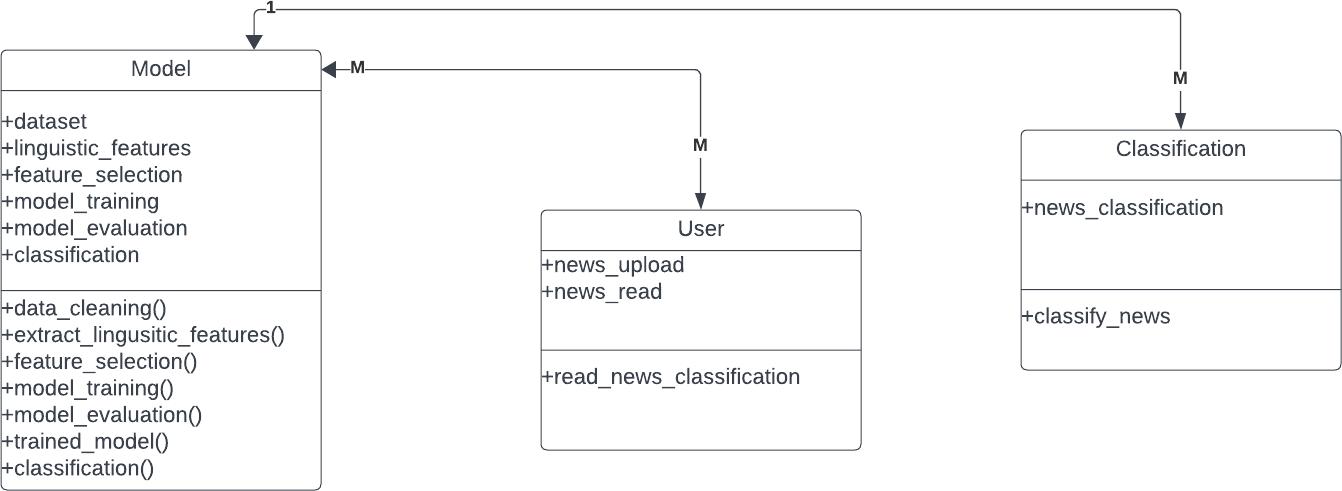


Figure 3.5: Class Diagram

# 3.6 system requirements

## Functional Requirements:

1. Text Analysis**:**

Parse and analyze textual content from news articles.

## Feature Extraction:

Extract relevant features from text, such as sentiment analysis.

## Machine Learning Model:

Implement a machine learning model for fake news detection.

## Real-time Detection:

Enable real-time analysis of news articles.

## Confidence Scoring:

Assign confidence scores to identified fake news.

## Fact-Checking Integration:

Integrate with credible fact-checking services.

## User Feedback:

Allow users to provide feedback on detected news items.

## Alerts and Notifications:

Generate alerts for potentially fake news.

## APIs:

Provide APIs for integration with other platforms.

Nonfunctional requirements

## Performance:

Provide real-time results.

Handle high volumes of articles concurrently.

## Reliability:

Minimize false positives and negatives. Ensure high availability.

## Scalability:

Scale to accommodate more users and articles.

## Security:

Protect user data and system integrity.

Implement secure authentication and authorization.

## Accuracy and Precision:

Achieve high accuracy.

Minimize false positives and negatives.

## Interoperability:

Ensure compatibility with various data formats.

## Maintainability:

Design with modularity and clear documentation.

## Usability:

Provide an intuitive interface.

## Compliance:

Comply with legal and ethical standards.

***10***-Resource Utilization:

Optimize resource usage.

***11***-Auditability:

Maintain detailed logs for auditing.

***12***-Internationalization:

Support multiple languages.

***13***-Ethical Considerations:

Implement safeguards against misuse.

Chapter 4

Dataset Model

## Introduction:

This chapter reviews Dataset information and setup of the dataset. The chapter is organized into: Dataset definition, Dataset properties.

Dataset Definition:

The dataset contains two types of articles fake and real News. This dataset was collected from real-world sources; the truthful articles were obtained by crawling articles from Reuters.com (News website). As for the fake news articles, they were collected from different sources. The fake news articles were collected from unreliable websites that were flagged by PolitiFact (a fact-checking organization in the USA) and Wikipedia. The dataset contains different types of articles on different topics; however, the majority of articles focus on political and World news topics.

The dataset consists of two CSV files. The first file named “True.csv” contains more than 12,600 articles from reuter.com. The second file named “Fake.csv” contains more than 12,600 articles from different fake news outlet resources. Each article contains the following information: article title, text, type, and the date the article was published on. To match the fake news data collected for kaggle.com, we focused mostly on collecting articles from 2016 to 2017. The data collected were cleaned and processed, however, the punctuations and mistakes that existed in the fake news were kept in the text.

The following table gives a breakdown of the categories and number of articles per category.

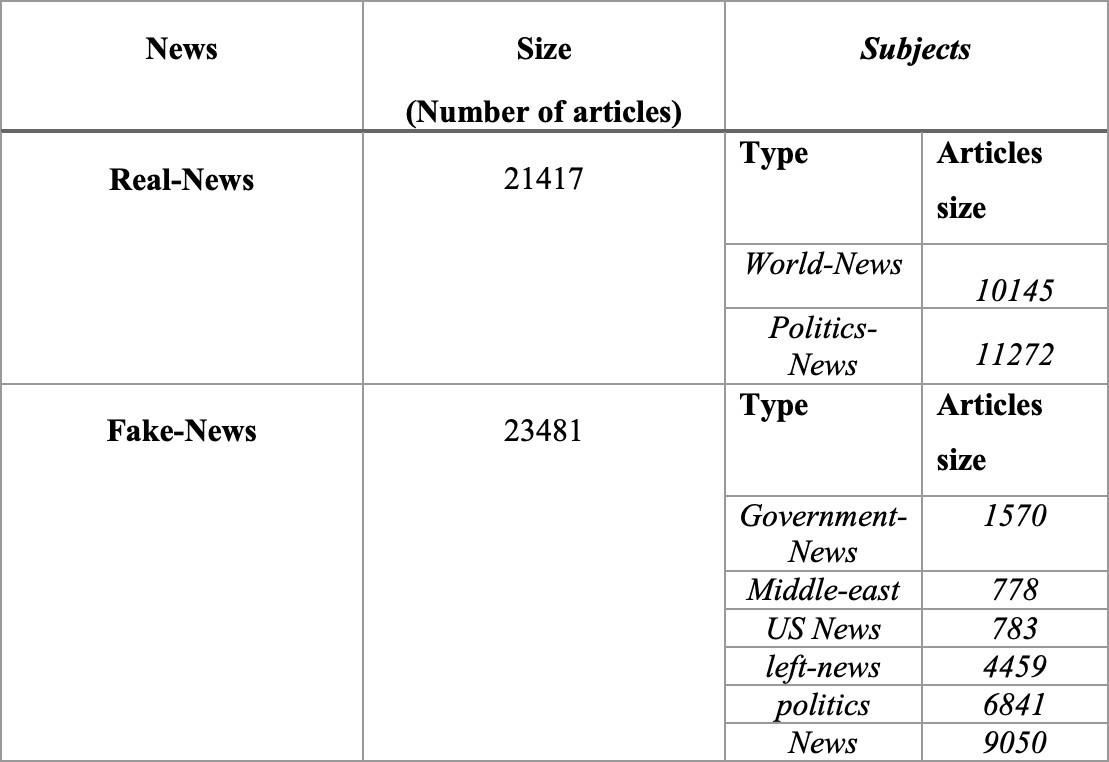


Figure 4: Dataset

Dataset Properties:

This Model is a compilation of several thousand fake news and truthful articles, sourced from various legitimate news sites and sites categorized as unreliable.

News is generally excluded in the Excel file, there are correct news and fake news. In each column there is a description of the news that is entered and name, and through detection methods in dataset origin of the news is determined.

Chapter 5

Proposed Model

In this chapter we will discuss:

System Architecture, text preprocessing, building model and news classification

**5.1 System Architecture**

A diagram of a decision model

Description automatically generated

System Architecture Figure (5.1)

Figure (5.1) explains the whole process of the system which is:

1. **News Dataset**: This is the initial dataset that contains a collection of news items. These could be articles, social media posts, or any other form of news content. The quality and diversity of this dataset can significantly impact the performance of the detection system.
2. **Preprocessing**: This step involves cleaning and transforming the raw news data into a format that can be understood by the machine learning model. This could involve removing irrelevant information, handling missing data, and converting text data into numerical features through techniques like tokenization, stemming, and vectorization.
3. **Features Dataset**: The preprocessed data is now a set of features that represent the original news items. These features could include the frequency of certain words, the length of the articles, the sentiment of the text, and so on.
4. **Splitting**: The Features Dataset is divided into a Training Dataset and a Testing Dataset. The Training Dataset is used to train the model, while the Testing Dataset is used to evaluate the model’s performance.
5. **Training**: In this phase, a machine learning model (the Decision Model) is trained using the Training Dataset. This involves feeding the features and their corresponding labels (i.e., whether the news item is fake or not) into the model, which learns to make predictions based on this data.
6. **Revision of Parameters**: If the model’s performance is not satisfactory, the parameters of the model are revised. This could involve adjusting the learning rate, the complexity of the model, the number of training iterations, and so on.
7. **Use**: Once the model’s parameters are acceptable, meaning the model performs well on the Training Dataset, it can be used to detect fake news.

**5.2 Text Preprocessing:**

This step is important because data from the real world is often messy and unstructured. It may contain errors, outliers, or irrelevant information that can negatively impact the performance of a model. Preprocessing helps clean up this data. So, there are several steps that need to be taken to get a clean (preprocessed) data:

1. **Dataset Loading:** Utilize the function pd.read\_csv('data.csv') to read a CSV file named ‘data.csv’ and store it in a DataFrame, referred to as ‘df’.
2. **Combining Columns:** The columns ‘Headline’ and ‘Body’ from the DataFrame are merged into a single column named ‘text’.
3. **Cleaning the Data:** The textual data is cleaned by transforming it to lowercase, eliminating punctuation, and substituting None values with an empty string.
4. **Tokenization:** The function word\_tokenize is used to divide the text into individual words, also known as “tokens”.
5. **Stopword Removal:** Words that are commonly used (such as ‘the’, ‘is’, ‘in’) and do not carry significant meaning are often eliminated from texts. This is accomplished using a list of stopwords from the nltk library.

A computer screen shot of text

Description automatically generated

6 .Lemmatization :  is a crucial step that involves reducing words to their base or dictionary form.

**Components :**

* Initializes the WordNetLemmatizer object, which will be used to convert words into their lemmas.

.

* Converts POS tags from the Penn Treebank format to a format compatible with WordNet for lemmatization purposes. It maps adjectives, verbs, nouns, and adverbs to their corresponding WordNet tags, defaulting to NOUN for unrecognized tags.

A computer screen shot of a program

Description automatically generated

* Accepts a list of tokens, retrieves their POS tags, and applies lemmatization to each token based on its POS tag. The lemmatized tokens are then joined into a single string.

A computer screen shot of a black background

Description automatically generated

* Applies the lemmatize\_text function to the ‘text’ column of the DataFrame. It ensures that the function is only applied to list objects, preserving the integrity of non-list data.



**Workflow :**

* The lemmatize\_text function is the core of this process. It takes a list of word tokens and performs the following steps:
  1. **POS Tagging:** Each token is assigned a POS tag using the pos\_tag function from NLTK.
  2. **Lemmatization:** The lemmatizer.lemmatize method is called with each word and its corresponding WordNet POS tag, obtained from the get\_wordnet\_pos function.
  3. **Reconstruction:** The lemmatized tokens are concatenated into a coherent string, ready for further analysis or vectorization.
* The final step integrates the lemmatization process into the DataFrame, ensuring that each text entry is appropriately preprocessed.

The next step will be splitting the dataset that has been preprocessed, Splitting the data into a training set and a testing set is a crucial step in machine learning. It helps evaluate the model’s performance on unseen data, providing a better understanding of the model’s ability to generalize

In this code, we’re partitioning your dataset into a training set and a testing set using the train\_test\_split function from sklearn.model\_selection. Here’s the breakdown of each component:

* df['text'] and df['Label']: These represent the features and labels of your dataset. The features (df['text']) serve as the inputs to your model, while the labels (df['Label']) are the predictions you want the model to make.
* test\_size=0.2: This indicates that 20% of the data will be allocated for the test set, with the remaining 80% designated for the training set.
* random\_state=42: This is utilized for reproducibility. By setting a specific random\_state, you ensure that the data is split consistently each time you execute the code. This can be beneficial when debugging or comparing models.

The function returns four variables:

* X\_train and y\_train: These are the features and labels for the training set. This data will be used to train your model.
* X\_test and y\_test: These are the features and labels for the test set. This data is not used during the training phase. Once the model is trained, you’ll use this data to assess the model’s ability to generalize to new, unseen data.

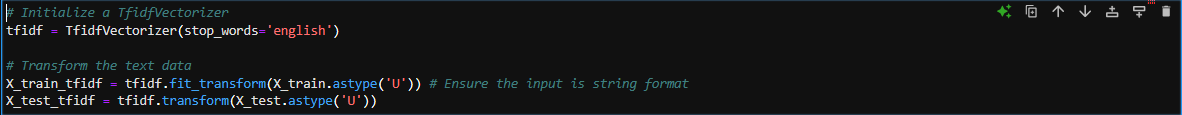
**

after that we use the TF-IDF vectorizer,  we use TF-IDF to convert textual data into a numerical form, which can be used as input for a model. It’s a way of representing text data for machine learning algorithms. The goal is to extract high-quality information from text and transform it into a format that algorithms can understand and use.

In this piece of code, you’re converting your text data into a numerical format that can be interpreted by a machine learning model. This is accomplished using the TfidfVectorizer from sklearn.feature\_extraction.text. Here’s an explanation of each part:

* tfidf = TfidfVectorizer(stop\_words='english'): This initializes a TfidfVectorizer, which will transform the text data into a matrix of TF-IDF features. The stop\_words='english' parameter instructs the vectorizer to disregard common English words like ‘the’, ‘is’, ‘in’, etc., which don’t provide much valuable information for model training.
* X\_train\_tfidf = tfidf.fit\_transform(X\_train.astype('U')): This line performs two tasks. The fit part learns the vocabulary of the training data (i.e., it identifies what words are used across all the documents). The transform part then uses this learned vocabulary to convert the text data into a document-term matrix, where each row represents a document and each column represents a word, and the value in each cell is the TF-IDF score of that word in that document. The astype('U') ensures that the input data is in Unicode string format, which is required by the TfidfVectorizer.
* X\_test\_tfidf = tfidf.transform(X\_test.astype('U')): This transforms the test data into a document-term matrix using the vocabulary learned from the training data. It’s important to note that we’re not fitting the vectorizer to the test data – we only transform it. This is because in a real-world scenario, our model will have to make predictions on unseen data, so it can only use the vocabulary it learned from the training data.

The output of these steps is X\_train\_tfidf and X\_test\_tfidf, which are the transformed versions of your training and testing data, respectively. These can now be used to train a machine learning model.



**5.3 Building Model:**

**XGBoost Hyperparameter Tuning and Model Saving**

This script performs hyperparameter tuning for an XGBoost classifier using randomized search and saves the best model to a file.

**Hyperparameters**

The param\_grid dictionary defines the hyperparameters for tuning. The keys are the hyperparameter names and the values are lists of possible values for each hyperparameter to  finding the right combination of them can significantly improve model accuracy, and here some of the hyperparameters used to tune the mode :

* learning\_rate: This is the step size at each iteration while moving toward a minimum of a loss function.
* max\_depth: This defines the maximum depth of a tree.
* n\_estimators: This is the number of trees you want to build before taking the maximum voting or averages of predictions.
* gamma: This is the minimum loss reduction required to make a further partition on a leaf node of the tree.
* subsample: This is the fraction of samples to be used for fitting the individual base learners.
* colsample\_bytree: This is the fraction of features to be used for each tree.
* reg\_alpha: This is the L1 regularization term on weights. It can be used to put a penalty on feature weights, effectively leading to feature selection.
* reg\_lambda: This is the L2 regularization term on weights. It encourages smoother weight values and can help prevent overfitting by penalizing peaky weights.
* **XGBoost Classifier Initialization**

An instance of the XGBoost classifier is created with the following parameters:

* use\_label\_encoder=False: Avoids using a label encoder.
* eval\_metric='logloss': Uses logarithmic loss as the evaluation metric during training.
* random\_state=42: Ensures reproducibility of results.

**Randomized Search Cross Validation**

An instance of RandomizedSearchCV is created to perform a randomized search on hyperparameters. The parameters for this instance are:

* clf: The base model for the random search.
* param\_distributions=param\_grid: The dictionary of hyperparameters to be tuned.
* n\_iter=20: The number of parameter settings that are sampled.
* scoring='accuracy': The method to evaluate the performance of the cross-validated model on the test set.
* n\_jobs=-1: The number of jobs to run in parallel. -1 means using all processors.
* cv=3: The cross-validation splitting strategy.
* verbose=3: The verbosity level.

**Model Training**

The fit method is called on the RandomizedSearchCV instance to train the model using the training data.

**Model Saving**

The trained model is saved to a file named 'model.pkl' using the joblib.dump function. This allows the model to be loaded later for further use, such as making predictions on new data.

A computer screen with many colorful text

Description automatically generated

**5.4 News Classification**

The Python script we’re about to discuss is a practical application of a variety of technologies. It’s a web-based application that uses a pre-trained model to classify news articles as real or fake. Let’s delve into the details of this interesting piece of code.

1. **Initialization**: The Flask application is initialized with app = Flask(\_\_name\_\_).



1. **Model Loading**: The pre-trained model (model.pkl) and the TF-IDF vectorizer (tfidf.pkl) are loaded using joblib.

A screen shot of a computer program

Description automatically generated

1. **Route Definition**: A route (/) is defined for both GET and POST requests. The function index () is executed when this route is accessed

**

1. **POST Request Handling**: If the request method is POST, the following steps are executed:
   * The news text is extracted from the form data (request.form['inputText']).
   * The news text is then preprocessed by converting it to lowercase, removing punctuation, and applying lemmatization while removing stopwords.
   * The preprocessed text is transformed into a TF-IDF vector (news\_text\_tfidf = tfidf.transform([news\_text])).
   * The transformed text is fed into the pre-trained model to predict whether the news is real or fake (news\_pred=random\_search.predict(news\_text\_tfidf)).
   * Depending on the prediction, the variables result and result\_class are set to “Real” or “Fake”.
   * The index.html template is rendered and returned with the predicted result.

**GET Request Handling**: If the request method is GET, the index.html template is rendered and returned without any result.



1. A computer screen shot of a program code

   Description automatically generated**Running the Application**
2. module), the Flask application is run with debugging turned off (app.run(debug=False).

*A black background with colorful text

Description automatically generated*

*A screenshot of a fake news detector

Description automatically generated*

Front-End Figure (5.4)

1. **Header and Title**:
   * At the top, there’s a red banner with white text that reads “F|D News.”
   * Below this, a smaller banner states, “The News Is Predicted as Fake News With.”
   * Centered prominently, larger bold text announces, “Fake News Detector.”
2. **Input Field and Button**:
   * Beneath the title, there’s an input box labeled “Enter the News Here…” where users can paste or type in news content.
   * Adjacent to the input field, there’s a red button labeled “Classification.” Users are likely to click this button to start the analysis.
3. **Result Display**:
   * At the bottom of the image, small black text indicates, “The result of news predicted is true/false.” This is presumably where the tool displays whether the analyzed news content is predicted as true or false.

Chapter 6

Experimental Results

This chapter reviews the model virtualization, model Testing, Validation and. compare the accuracy of our model to other models.

6.1 Confusion Matrix

A computer screen shot of a program code

Description automatically generated

A screenshot of a graph

Description automatically generated

Figure (6.1)

6.2 Classification Report

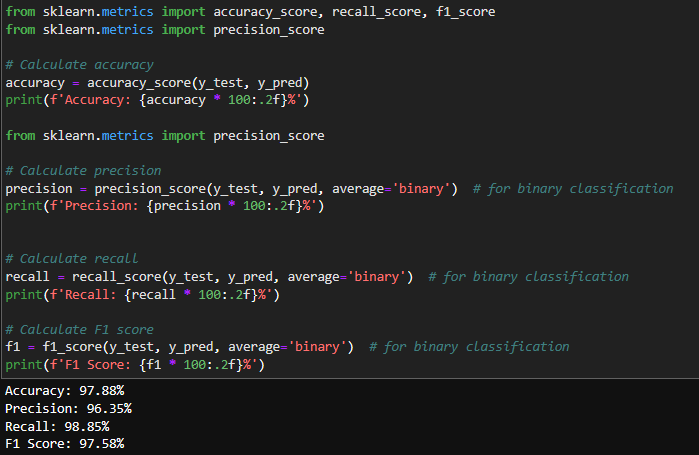


Figure (6.2)

6.3 RandomSearchCV and XGBoost

A screen shot of a computer

Description automatically generated

Figure (6.3)

6.4 Comparison between the proposed model and other models:

A graph of different colored bars

Description automatically generated

Figure (6.4)

Figure (6.4) is a graphical representation of average performance of learning algorithms on all datasets using precision, recall, and F1-score. There is not much difference between the performance of learning algorithms using various performance metrics except for linear SVM, KNN, Wang-CNN, and Wang-Bi-LSTM. e ensemble learner XGBoost performed better in comparison to other learning models on all performance, the ensemble learner XGBoost performed better in comparison to other learning models on all performance metrics. The main factor leading to the superior performance of XGBoost is the working principle which efficiently identifies errors and minimizes them in each iteration.

Chapter 7

Conclusion and Future work

7.1 Conclusion:

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. In this research, we discussed the problem of classifying fake news articles using machine learning models and ensemble techniques.

The data we used in our work is collected from the World Wide Web and contains news articles from various domains to cover most of the news rather than specifically classifying political news.

The primary aim of the research is to identify patterns in text that differentiate fake articles from true news. We extracted different textual features from the articles using an LIWC tool and used the feature set as an input to the models.

The learning models were trained and parameter-tuned to obtain optimal accuracy, some models have achieved comparatively higher accuracy than others.

We used multiple performance metrics to compare the results for each algorithm. The ensemble learners have shown an overall better score on all performance metrics as compared to the individual learners.

7.2 Future Work:

Fake news detection has many open issues that require attention of researchers. For instance, to reduce the spread of fake news, identifying key elements involved in the spread of news is an important step. Graph theory and machine learning techniques can be employed to identify the key sources involved in spread of fake news. Likewise, real time fake news identification in videos can be another possible future direction.

Chapter 8

References

<https://www.hindawi.com/journals/complexity/2020/8885861/>

-----------------------------------------------------------------------------

<https://www.politifact.com/>

-----------------------------------------------------------------------------

<https://www.snopes.com/fact-check/>

-----------------------------------------------------------------------------

<https://arxiv.org/abs/2102.04458>

-----------------------------------------------------------------------------

<https://www.kaggle.com/datasets/jruvika/fake-news-detection>

-----------------------------------------------------------------------------

<https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset>

-----------------------------------------------------------------------------

<https://www.nvidia.com/en-us/glossary/data-science/xgboost/>

------------------------------------------------------------------------------

<https://en.wikipedia.org/wiki/Ensemble_learning>

-----------------------------------------------------------------------------

<https://developer.nvidia.com/blog/gradient-boosting-decision-trees-xgboost-cuda/>

-----------------------------------------------------------------------------

<https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/>

-----------------------------------------------------------------------------

<https://medium.com/@techynilesh/xgboost-algorithm-explained-in-less-than-5-minutes-b561dcc1ccee>

-----------------------------------------------------------------------------

<https://www.geeksforgeeks.org/xgboost/>

------------------------------------------------------------------------------

<https://www.datacamp.com/tutorial/xgboost-in-python>

-----------------------------------------------------------------------------

<https://www.kaggle.com/code/soheiltehranipour/xgboost-tutorial-classification>

-----------------------------------------------------------------------------

<https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets>

-----------------------------------------------------------------------------