

FAKE NEWS DETECTION USING MACHINE LEARNING

Submitted in partial fulfilment of the requirements
for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By
Govindareddypalli Chandu(38110172)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH – 2022



SATHYABAMA



INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Govindareddypalli Chandu(38110172)** who carried out the project entitled “**Fake News Detection Using Machine Learning Approach**” under my supervision from November 2020 to March 2021.

Internal Guide

Dr.J.Refonaa

Head of the Department

Dr. S. Vigneshwari M.E., Ph.D., and Dr. L. Lakshmanan M.E., Ph.D.,

Submitted for Viva voce Examination held on ____

Internal Examiner

External Examiner

DECLARATION

I **Govindareddypalli Chandu (Reg No:38110172)** hereby declare that the Project Report entitled “**Fake News Detection Using Machine Learning Approach**” done by us under the guidance of **Dr.J.Refonaa** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in 2021-2022.

DATE:

**PLACE:
CANDIDATE**

SIGNATURE OF THE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E.,Ph.D.**, Dean, School of Computing **Dr.S.Vigneshwari M.E., Ph.D.** and **Dr.L.Lakshmanan M.E., Ph.D.** , Heads of the Department of Computer Science and Engineering for providing us necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and a deep sense of gratitude to my Project Guide **Dr.j.Refonaa**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The purpose of this project is to work on the fake news dataset and to detect the fake news. This project gives an overview of some of the most popular machine learning algorithms- Naive Bayes, SVM, K-NN and Logistic Regression. The news is categorized as fake, real, satire, bias or hate based on two different datasets. The term 'Fake News' is used here to define the unlike problems from fake news to satire news or the hate news. Online fake news has been a topic of interest these days and has been used in multitude of ways. I will be using machine learning algorithms in Python with Scikit-Learn to train and test the data to find out the accuracies of each algorithm. Also, the description of the algorithms is presented and comparison of their performance to find out which algorithm is suitable for this kind of text dataset.

Keywords: *dataset, Machine Learning-Regression methods, mean absolute error, R2-score*

TABLE OF CONTENTS

Chapter No	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF ABBREVIATIONS	viii
	LIST OF FIGURES	ix
1	INTRODUCTION	1
	1.1 DOMAIN OVERVIEW	1
2	LITERATURE SURVEY	4
	2.1. GENERAL	4
	2.2. REVIEW OF LITERATURE SURVEY	4
3	METHODOLOGY	10
	3.1. EXISTING SYSTEM	10
	3.2. PROPOSED SYSTEM	10
	3.3. OBJECTIVE	10
	3.4. SCOPE	10
	3.5. SOFTWARE AND HARDWARE REQUIREMENTS	11
	3.5.1. SOFTWARE REQUIREMENTS	11
	3.5.2. HARDWARE REQUIREMENTS	11
	3.5.3. PROJECT REQUIREMENTS	11
	3.5.3.1. FUNCTIONAL REQUIREMENTS	11
	3.5.3.2. NON-FUNCTIONAL REQUIREMENTS	11
	3.6. SOFTWARE DESCRIPTION	11
	3.6.1. CONDA	12
	3.6.2. THE JUPYTER NOTEBOOK	12
	3.6.3. NOTEBOOK DOCUMENT	12
	3.6.4. JUPYTER NOTEBOOK APP	12
	3.6.5. KERNEL	12
	3.6.6. NOTEBOOK DASHBOARD	13
	3.7. SYSTEM ARCHITECTURE	13
	3.8. MODULES	13

	3.8.1. VARIABLE IDENTIFICATION PROCESS / DATA VALIDATION PROCESS AND DATA CLEANING	13
	3.8.1.1.DATA CLEANING	14
	3.8.2. EXPLORATION DATA ANALYSIS OF VISUALIZATION AND NORMALIZATION	15
	3.8.2.1. EXPLORATION DATA ANALYSIS OF VISUALIZATION	15
	3.8.2.2. DATA NORMALIZATION	16
	3.8.3. TRAINING A MODEL BY GIVEN ATTRIBUTES AND USING RANDOM FOREST ALGORITHM	17
	3.8.3.1. RANDOM FOREST	17
	3.8.4. PERFORMANCE MEASUREMENTS OF KNN AND DECISION TREE	18
	3.8.4.1. KNN	18
	3.8.4.2. DECISION TREE	18
	3.8.5. PERFORMANCE MEASUREMENTS OF LASSO AND LINEAR REGRESSION	19
	3.8.5.1. LINEAR REGRESSION	19
	3.8.5.2. LASSO REGRESSION	20
	3.8.6. GUI BASED PREDICTION OF FAKE NEWS YIELD	20
	3.8.7. PARAMETER CALCULATIONS	20
	3.8.7.1. MEAN SQUARED ERROR	20
	3.8.7.2. ROOT MEAN SQUARED ERROR	21
	3.8.7.3. MEAN ABSOLUTE ERROR	21
	3.8.7.4. R2-SCORE	22
4	RESULTS AND DISCUSSIONS	23
	4.1. GRAPH COMPARING MSE VALUES IN VARIOUS ALGORITHM	23
	4.2. GRAPH COMAPARING R2_SCORE VALUES IN VARIOUS ALGORITHM	23
	4.3. GRAPH COMAPARING MAE VALUES IN VARIOUS ALGORITHM	24
	4.4. GRAPH COMPARING RMSE VALUES IN VARIOUS ALGORITHM	24
	4.5. OUTPUTS	24
5	CONCLUSION	26
	5.1. FUTURE WORK	26
	REFERENCES	27
	SOURCE CODE	28
	PLAGARISM REPORT	45
	PAPER	46

LIST OF ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligent
DA	Data Analytic
MSE	Mean Squared Error
RMSE	Root Mean Squared Error

LIST OF FIGURES

Figure no	Figure name	Page no
1.1	Process of Machine Learning	2
3.1	System Architecture	13
3.2	Given Data Frame	14
3.3	Cleaning of Dataset	15
3.4	Percentage Level of Data in Different Years	16
3.5	Removing Outliers	16
3.6	Splitting A Data Set in To Training and Training Data	17
4.1	When Correct News Is Given	25
4.2	When Fake News Is Given	25

CHAPTER 1

INTRODUCTION

Social media for news consumption is a double-edged sword. On the one hand, its low cost, easy access, and rapid dissemination of information lead people to seek out and consume news from social media. On the other hand, it enables the wide spread of “fake news”, i.e., low quality news with intentionally false information. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention.

Fake news detection on social media presents unique characteristics and challenges that make existing detection algorithms from traditional news media ineffective or not applicable. First, fake news is intentionally written to mislead readers to believe false information, which makes it difficult and nontrivial to detect based on news content; therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a determination. Second, exploiting this auxiliary information is challenging in and of itself as users’ social engagements with fake news produce data that is big, incomplete, unstructured, and noisy.

1.1 DOMAIN OVERVIEW:

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based

on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modelling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be

bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Fig 1.1 Process of Machine learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class

classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

There exists a large body of research on the topic of machine learning methods for deception detection, most of it has been focusing on classifying online reviews and publicly available social media posts. Particularly since late 2016 during the American Presidential election, the question of determining 'fake news' has also been the subject of particular attention within the literature.

Conroy, Rubin, and Chen outlines several approaches that seem promising towards the aim of perfectly classify the misleading articles. They note that simple content-related n-grams and shallow parts-of-speech tagging have proven insufficient for the classification task, often failing to account for important context information. Rather, these methods have been shown useful only in tandem with more complex methods of analysis. Deep Syntax analysis using Probabilistic Context Free Grammars have been shown to be particularly valuable in combination with n-gram methods. Feng, Banerjee, and Choi are able to achieve 85%-91% accuracy in deception related classification tasks using online review corpora.

CHAPTER 2

LITERATURE SURVEY

2.1 GENERAL

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

2.2 REVIEW OF LITERATURE SURVEY:

Title: The Definitional Challenges of Fake News

Author: Haiden,L..., & Althuis, j

Year : (2018).

Description:

Rapid determination of soil organic matter (SOM) using regression models based on soil reflectance spectral data serves an important function in precision agriculture. “Deviation of arch” (DOA)-based regression and partial least squares regression (PLSR) are two modelling approaches to predict SOM. However, few studies have explored the accuracy of the DOA

based regression and PLSR models. Therefore, the DOA-based regression and PLSR were applied to the visible near-infrared (VNIR) spectra to estimate SOM content in the case of various dataset divisions. A two-fold cross-validation scheme was adopted and repeated 10000 times for rigorous evaluation of the DOA-based models in comparison with the widely used PLSR model. Soil samples were collected for SOM analysis in the coastal area of northern Jiangsu Province, China. The results indicated that both modelling methods provided reasonable estimation of SOM, with PLSR outperforming DOA-based regression in general. However, the performance of PLSR for the validation dataset decreased more noticeably. Among the

four DOA-based regression models, a linear model provided the best estimation of SOM and a cut-off of SOM content (19.76 g kg⁻¹), and the performance for calibration and validation datasets was consistent. As the SOM content exceeded 19.76 g kg⁻¹, SOM became more effective in masking the spectral features of other soil properties to a certain extent. This work confirmed that reflectance spectroscopy combined with PLSR could serve as a non-destructive and cost-efficient way for rapid determination of SOM when

hyper spectral data were available. The DOA-based model, which requires only 3 bands in the visible spectra, also provided SOM estimation with acceptable accuracy.

Title: Media-Rich Fake News Detection: A Survey

Author: Shivam B. Parikh and Pradeep K. Atrey /2018

Year: 2018

Description:

Fake News has been around for decades and with the advent of social media and modern day journalism at its peak, detection of media-rich fake news has been a popular topic in the research community. Given the challenges associated with detecting fake news research problem, researchers around the globe are trying to understand the basic characteristics of the problem statement.

Linguistic Features based Methods, Deception Modeling based Methods, Clustering based Methods, Predictive Modeling based Methods, Content Cues based Methods, Non-Text Cues based Methods.

Although this form of method is often deemed to be better than cue-based methods it unfortunately still does not extract and fully exploit the rich semantic and syntactic information in the content.

Title: Automatic Deception Detection: Methods for Finding Fake News

Author: Niall J. Conroy, Victoria L. Rubin, and Yimin Chen

Year: 2015

Description:

This research surveys the current state-of-the-art technologies that are instrumental in the adoption and development of fake news detection. "Fake

news detection” is defined as the task of categorizing news along a continuum of veracity, with an associated measure of certainty. Veracity is compromised by the occurrence of intentional deceptions. There are two potential limitations in this method: (Ciampiaglia et al., 2015)

attributes and descriptors depends on a sufficient amount of mined content for profiles, and the challenge of correctly associating descriptors with extracted attributes.

Title: Weakly supervised learning for fake news detection on Twitter

Author: Stefan Helmstetter Heiko Paulheim

Year: 2018

Description:

The problem of automatic detection of fake news in social media, e.g., on Twitter, has recently drawn some attention. Although, from a technical perspective, it can be regarded as a straight-forward, binary classification problem, the major challenge is the collection of large enough training corpora, since manual annotation of tweets as fake or non-fake news is an expensive and tedious endeavor.

In this paper, we consider the novel problem of unsupervised fake news detection. We extract the social media users’ opinions from their hierarchy social engagement information. By treating the truths of news and the credibility of users as latent random variables, a probabilistic graphical model is built to capture the complete generative spectrum.

Title: Fake News Detection in Social Media

Author: Kai Shu and Huan Liu

Year: 2018

Description:

In the past decade, social media has become increasingly popular for news consumption due to its easy access, fast dissemination, and low cost. However, social media also enables the wide propagation of “fake news,” i.e., news with intentionally false information. Fake news on social media can

have significant negative societal effects. Common assimilation methods face some difficulties due to the scarce, constant, or similar nature of the input parameters. For example, yield spatial heterogeneity simulation, coexistence of common assimilation methods and the nutrient module, and time cost are relatively important limiting factors. To address the yield simulation problems at field scale, a simple yet effective method with fast algorithms is presented for assimilating the time-series HJ-1 A/B data into the WOFOST model in order to improve the spring maize yield simulation. First, the WOFOST model is calibrated and validated to obtain the precise mean yield. Second, the time-series leaf area index (LAI) is calculated from the HJ data using an empirical regression model. Third, some fast algorithms are developed to complete assimilation. Existing datasets have some limitations that we try to address in our data repository. For example, Buzz Feed News only contains headlines and text for each news piece and covers news articles from very few news agencies.

Title: Automatic Online Fake News Detection Combining Content and Social Signals

Author: Della Vedova, M. L., Tacchini, E., Moret, S., Ballarin, G., DiPierro, M., & de Alfaro, L.

Year: 2018

Description:

The proliferation and rapid diffusion of fake news on the Internet highlight the need of automatic hoax detection systems. In the context of social networks, machine learning (ML) methods can be used for this purpose

This empirical regression model that was developed in Kansas was successfully applied directly in Ukraine. The model forecast winter wheat production in Ukraine six weeks prior to harvest with a 10% error of the official production numbers. In 2009 the model was run in real-time in Ukraine and forecast production within 7% of the official statistics which were released after the harvest. The same regression model forecast winter wheat production in Ukraine within 10% of the official reported production numbers six weeks prior to harvest. Using new data from MODIS, this method is simple, has limited data requirements, and can provide an indication of winter wheat production shortfalls and surplus prior to harvest in regions where minimal ground data is available. We then tested the chatbot with a completely independent

set of news, whose content was not part of the three previously mentioned datasets

Title: Machine Learning Approaches to detect fake new from social media

Author: Kim, NariLee, Yang-Won

Year: 2016

Description:

Machine learning, which is an efficient empirical method for classification and prediction, Social media for news consumption is a double-edged sword. On the one hand, its low cost, easy access, and rapid dissemination of information lead people to seek out and consume news from social media. On the other hand, it enables the wide spread of “fake news”, i.e., low quality news with intentionally false information. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society.

Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention. Fake news detection on social media presents unique characteristics and challenges that make existing detection algorithms from traditional news media ineffective or not applicable. First, fake news is intentionally written to mislead readers to believe false information, which makes it difficult and nontrivial to detect based on news content; therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a determination.

And international Fake News trade. Because South Korea highly depends on imports of most major grains except for rice, reasonable estimations of Fake News yields are more required under recent conditions of climate changes and various disasters. Remote sensing data has been widely used in the estimation of Fake News yields by employing statistical methods such as regression model. It conducted multivariate regression analyses to estimate corn and soybean yields in Iowa using MODIS (Moderate Resolution Imaging Spector radiometer) NDVI (Normalized Difference Vegetation Index), climate factors and soil moisture and presented regression models for the estimation of winter wheat yields using MODIS NDVI and weather data in Shandong, China. It estimated corn and soybean yields using several MODIS products and climatic variables for Midwestern United States (US) and represented prediction errors of about 10 %. It built multiple regression models using MODIS NDVI and weather data to estimate rice yields in North Korea and showed the RMSE of 0.27 ton/ha. Most of the previous studies are based on the multivariate regression analysis using the relationship between Fake News yields and agro-environmental factors such as vegetation index, climate variables and soil properties.

The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention. Fake news detection on social media presents unique characteristics and challenges that make existing detection algorithms from traditional news media ineffective or not applicable. First, fake news is intentionally written to

mislead readers to believe false information, which makes it difficult and nontrivial to detect based on news content; therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a determination. Second, exploiting this auxiliary information is challenging in and of itself as users' social engagements with fake news produce data that is big, incomplete, unstructured, and noisy.

CHAPTER 3

METHODOLOGY

3.1 EXISTING SYSTEM

There exists a large body of research on the topic of machine learning methods for deception detection, most of it has been focusing on classifying online reviews and publicly available social media posts. Particularly since late 2016 during the American Presidential election, the question of determining 'fake news' has also been the subject of particular attention within the literature.

Conroy, Rubin, and Chen outlines several approaches that seem promising towards the aim of perfectly classify the misleading articles. They note that simple content-related n-grams and shallow parts-of-speech tagging have proven insufficient for the classification task, often failing to account for important context information.

3.2 PROPOSED SYSTEM

- In this paper a model is build based on the count vectorizer or a tfidf matrix word tallies relatives to how often they are used in other artices in your dataset) can help .
- Since this problem is a kind of text classification, Implementing a Naive Bayes classifier will be best as this is standard for text-based processing.
- The actual goal is in developing a model which was the text transformation (count vectorizer vs tfidf vectorizer) and choosing which type of text to use (headlines vs full text).

STEPS

- Data Use:
- Preprocessing:
- Feature Extraction:
- Training the Classifier

3.3 OBJECTIVE

- Data validation
- Data Cleaning/ Preparing
- Data Visualization
- Using algorithms (Support Vector Machine- SVM, Naive Bayes, Logistic Regression, K Nearest Neighbors)

3.4 SCOPE

The objective of this project is to examine the problems and possible significances related with the spread of fake news. We will be working on

different fake news data set in which we will apply different machine learning algorithms to train the data and test it to find which news is the real news or which one is the fake news. As the fake news is a problem that is heavily affecting society and our perception of not only the media but also facts and opinions themselves

3.5 HARDWARE AND SOFTWARE REQUIREMENTS

3.5.1 Software Requirements

Operating System: Windows

Tool: Anaconda with Jupyter Notebook

3.5.2 Hardware requirements

Processor: Pentium IV/III

Hard disk: minimum 80 GB

RAM: minimum 2 GB

3.5.3 PROJECT REQUIREMENTS

3.5.3.1 Functional requirements

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

3.5.3.2 Non-Functional Requirements

- Process of functional steps,
- Problem define
- Preparing data
- Evaluating algorithms
- Improving results
- Prediction the result

3.6 SOFTWARE DESCRIPTION

Anaconda is a free and open distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands.

Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

3.6.1 Conda

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-languages. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository

3.6.2 The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

3.6.3 Notebook Document

Notebook documents (or “notebooks”, all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

3.6.4 Jupyter Notebook App

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

3.6.5 Kernel

A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results. Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down.

3.6.6 Notebook Dashboard

The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown). The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files.

3.7 System Architecture

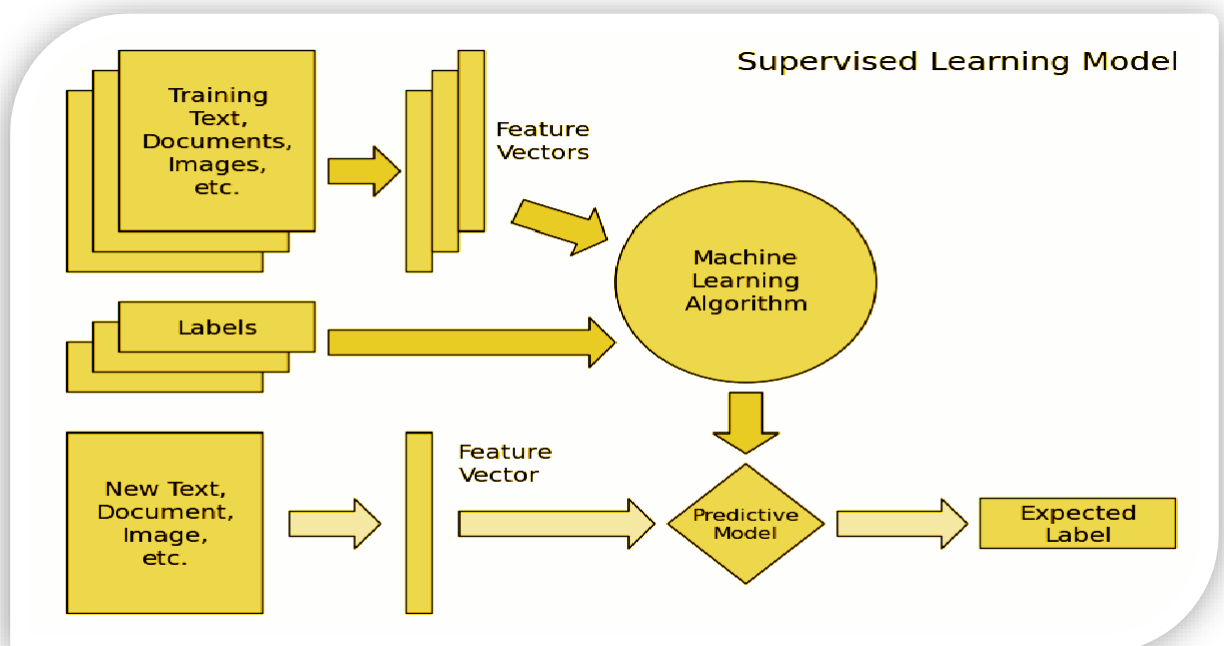


Fig 3.7 system architecture

3.8 Modules

- Data validation and Data cleaning (Module 1)
- Exploration data analysis of visualization and normalizing the data (Module 2)
- Training a model by given attributes and using Random Forest algorithm (Module 3)
- Performance measurements of KNN and Decision Tree (Module-05)

- Performance measurements of Linear and Lasso Regression (Module-05)
- GUI based prediction of Fake News yield (Module-06)

3.8.1 Module-01: Variable Identification Process / data validation process and Data cleaning:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the

training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers uses this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model. For example, time series data can be analysed by regression algorithms; classification algorithms can be used to analyse discrete data. (For example, to show the data type format of given dataset)

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0

Fig 3.2 Given Data Frame

3.8.1.1. Data cleaning:

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back

from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

```
: df_input = df_input[df_input['Area'] > 0 ]
df_input = df_input[df_input['Production'] > 0 ]
df_input = df_input.replace(r'^\s*$', np.NaN, regex=True)
df_input = df_input.dropna()
df_input["ProductionPerArea"] = ((df_input["Production"])/(df_input["Area"]))
df_input = df_input.drop(columns=['State_Name', 'Area', 'Production'])
```

Fig 3.3: Cleaning the Dataset

3.8.2 Module-02: Exploration data analysis of visualization and normalization:

3.8.2.1 Exploration data analysis of visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end. Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and inapplied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data. How to chart time series data with line plots and categorical quantities with bar charts. How to summarize data distributions with histograms and box plots. How to summarize the relationship between variables with scatter plots.

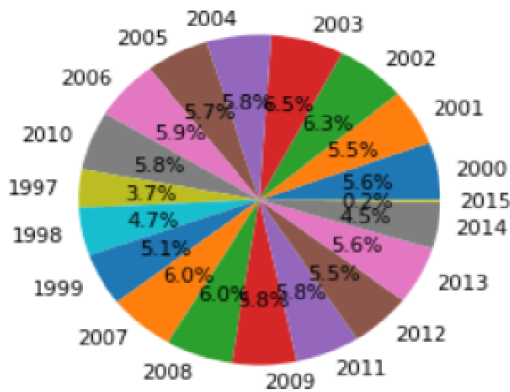


Fig 3.4 Percentage Level Of Data In Different Years

Many machine learning algorithms are sensitive to the range and distribution of attribute values in the input data. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results. Even before predictive models are prepared on training data, outliers can result in misleading representations and in turn misleading interpretations of collected data. Outliers can skew the summary 22 distribution of attribute values in descriptive statistics like mean and standard deviation and in plots such as histograms and scatterplots, compressing the body of the data. Finally, outliers can represent examples of data instances that are relevant to the problem such as anomalies in the case of fraud detection and computer security.

It couldn't fit the model on the training data and can't say that the say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

```
p1 = np.percentile(np.array(df_input['ProductionPerArea']), 25)
p2 = np.percentile(np.array(df_input['ProductionPerArea']), 99)
df_input = df_input[df_input['ProductionPerArea'] > p1]
df_input = df_input[df_input['ProductionPerArea'] < p2]
```

Fig 3.5 Removing outliers

3.8.2.2 Data Normalization

To train data with string datatype we use label encoder to convert them to numeric data. Label encoding simply assigns each data point a number starting from .the attributes we are label encoding is district name, season and Fake News . Then to scale each data attribute we use two types of scaling techniques which are

MinMax transform and normalization. The attributes district name, season, Fake News is scaled by MinMax transform and production per area is scaled by normalization. MinMax transform is done by subtracting each datapoint from minimum among datapoints then dividing by maximum- minimum. normalization is done by dividing datapoint with standard deviation. We choose normalization to productionperarea as many data points are low and min max transform is yielding good results.

3.8.3 Module 3: Training a model by given attributes and using Random Forest algorithm

At First, we import all the required modules. Then we use appropriate cleaning methods to clean our data. Then we split our data according to the required output. We split our dataset into train and test data using train_test_split technique. The X prefix indicates the element esteems and y prefix indicates target esteems. This splits the dataset into train and test information in the proportion which we mentioned. Here we are using a proportion of 80:20. At this point we epitomize any calculation. Here we fit our preparation information into this calculation so our system can get prepared utilizing this information. Here the training part is finished

```
from sklearn.model_selection import train_test_split
df_small = df_input
df_small.columns.name = None
df=df_small
x_train, x_test, y_train, y_test =
train_test_split(df.iloc[:, :-1], df.iloc[:, -1], test_size=0.2, random_state=2)
```

Fig 3.6 Splitting A Data Set in To Training And Training Data

3.8.3.1 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks. The following are the basic steps involved in performing the random forest algorithm

pick N random records from the dataset. Build a decision tree based on these N records. Choose the number of trees you want in your algorithm and repeat steps

1 and 2. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

3.8.4 Module 4 :Performance measurements of kNN and Decision Tree

3.8.4.1 KNN

K-Nearest Neighbor (KNN) K-Nearest Neighbor is a supervised machine learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors) and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors. In the distance-weighted nearest neighbor algorithm, it weights the contribution of each of the k neighbors according to their distance using the following query giving greater weight to the closest neighbors.

Usually KNN is robust to noisy data since it is averaging the k-nearest neighbors. The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labeled points and uses them to learn how to label other points. To label a new point, it looks at the labeled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the “k” is the number of neighbors it checks). Makes predictions about the validation set using the entire training set. KNN makes a prediction about a new instance by searching through the entire set to find the k “closest” instances. “Closeness” is determined using a proximity measurement (Euclidean) across all features.

3.8.4.2 Decision Tree

It is one of the most powerful and popular algorithms. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

Assumptions of Decision tree:

- At the beginning, we consider the whole training set as the root.
- Attributes are assumed to be categorical for information gain, attributes are assumed to be continuous.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or internal node.

Decision tree builds classification or regression models in the form of a tree

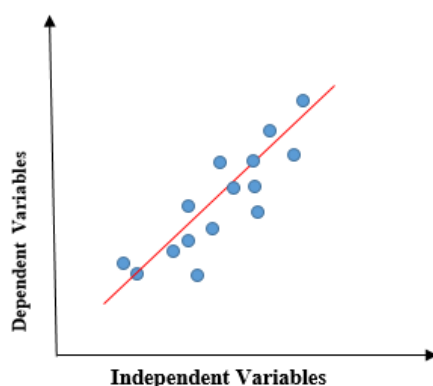
structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers

3.8.5 Module 5: Performance measurements of Lasso and Linear Regression

3.8.5.1 Logistic Regression

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression. *If there is a single input variable (x), such linear regression is called simple linear regression. And if there is more than one input variable, such linear regression is called multiple linear regression.* The linear regression model gives a sloped straight line describing the relationship within the variables.



The above graph presents the linear relationship between the dependent variable and independent variables. When the value of x (independent variable) increases, the value of y (dependent variable) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best.

3.8.5.2 Lasso Regression

Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

Lasso Regression uses L1 regularization technique (will be discussed later in this article). It is used when we have a greater number of features because it automatically performs feature selection

3.8.6 Module 6: GUI based prediction of Fake News yield

- we created a webpage and deployed it in Heroku. we used HTML, CSS and js for the frond end.
- In the backend we used python language and flask framework to implement

3.8.7 Parameter calculations

Regression refers to predictive modelling problems that involve predicting a numeric value.

It is different from classification that involves predicting a class label. Unlike classification, you cannot use classification accuracy to evaluate the predictions made by a regression model.

Instead, you must use error metrics specifically designed for evaluating predictions made on regression problems.

There are four error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

- Mean Squared Error (MSE).
- Root Mean Squared Error (RMSE).
- Mean Absolute Error (MAE)
- R2_score

3.8.7.1 Mean Squared Error

Mean Squared Error, or MSE for short, is a popular error metric for regression problems. It is also an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. Here "*least squares*" refers to minimizing the mean squared error between predictions and expected values.

The MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset.

- $MSE = 1 / N * \sum \text{for } i \text{ to } N (y_i - \hat{y}_i)^2$

Where y_i is the i 'th expected value in the dataset and $\hat{y}_i$ is the i 'th predicted value. The difference between these two values is squared, which has the effect of removing the sign, resulting in a positive error value.

The squaring also has the effect of inflating or magnifying large errors. That is, the larger the difference between the predicted and expected values, the larger the resulting squared positive error. This has the effect of “*punishing*” models more for larger errors when MSE is used as a loss function. It also has the effect of “*punishing*” models by inflating the average error score when used as a metric.

3.8.7.2 Root Mean Squared Error:

RMSE, is an extension of the mean squared error. Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

For example, if your target variable has the units “*dollars*,” then the RMSE error score will also have the unit “*dollars*” and not “*squared dollars*” like the MSE.

As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.

The RMSE can be calculated as follows:

- $RMSE = \sqrt{1 / N * \sum \text{for } i \text{ to } N (y_i - \hat{y}_i)^2}$

Where y_i is the i 'th expected value in the dataset, $\hat{y}_i$ is the i 'th predicted value, and $\sqrt{}$ is the square root function.

We can restate the RMSE in terms of the MSE as:

- $RMSE = \sqrt{MSE}$

Note that the RMSE cannot be calculated as the average of the square root of the mean squared error values. This is a common error made by beginners and is an example of Jensen's inequality

3.8.7.3 Mean Absolute Error

MAE, is a popular metric because, like RMSE, the units of the error score match the units of the target value that is being predicted.

Unlike the RMSE, the changes in MAE are linear and therefore intuitive.

That is, MSE and RMSE punish larger errors more than smaller errors, inflating or magnifying the mean error score. This is due to the square of the error value. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

As its name suggests, the MAE score is calculated as the average of the absolute error values. Absolute or $\text{abs}()$ is a mathematical function that simply makes a number positive. Therefore, the difference between an expected and predicted

value may be positive or negative and is forced to be positive when calculating the MAE.

The MAE can be calculated as follows:

- $MAE = 1 / N * \sum_{i=1}^N \text{abs}(y_i - \hat{y}_i)$

Where y_i is the i 'th expected value in the dataset, $\hat{y}_i$ is the i 'th predicted value and $\text{abs}()$ is the absolute function

3.8.7.4 R2_score:

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second

$$R^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}}$$

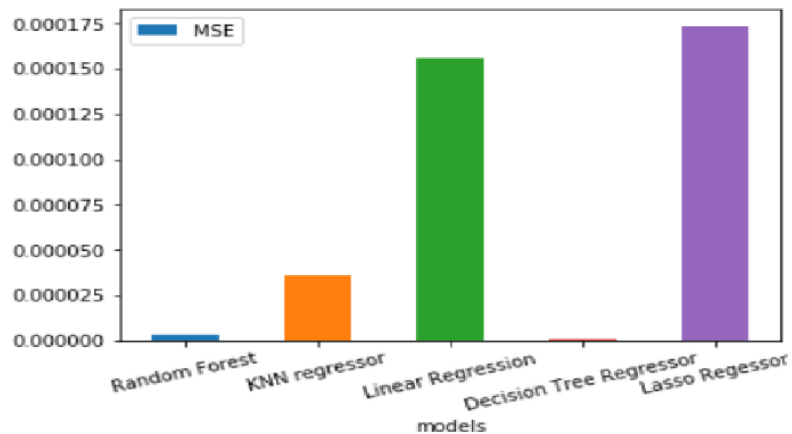
variable

CHAPTER 4

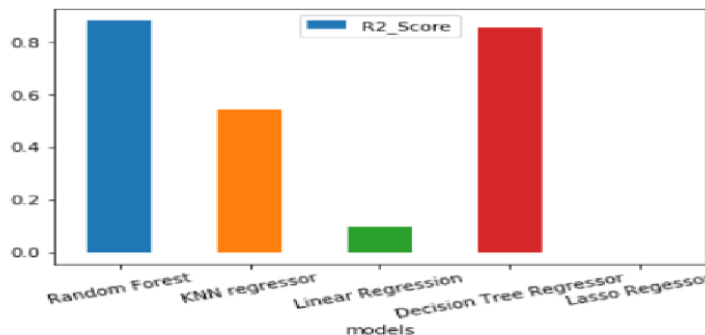
RESULTS AND DISCUSSION

	MSE	MAE	RMSE	R2_Score	models
0	3.426823e-06	0.001505	0.004558	0.880158	Random Forest
1	3.595377e-05	0.003637	0.008913	0.541729	KNN regressor
2	1.557740e-04	0.006636	0.012498	0.098844	Linear Regression
3	1.761657e-07	0.001490	0.005049	0.852930	Decision Tree Regressor
4	1.737817e-04	0.007141	0.013166	-0.000002	Lasso Regressor

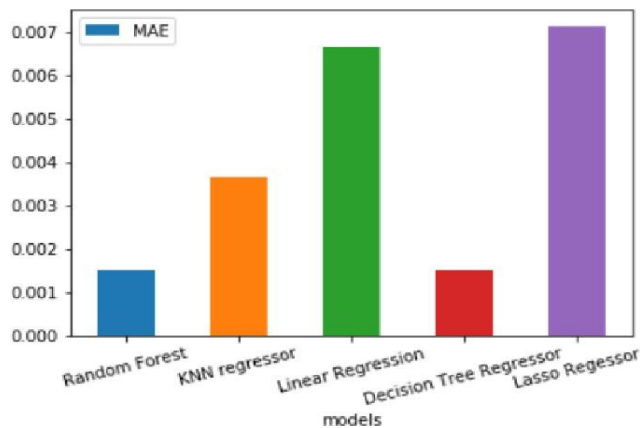
4.1. Graph comparing MSE values in various algorithms



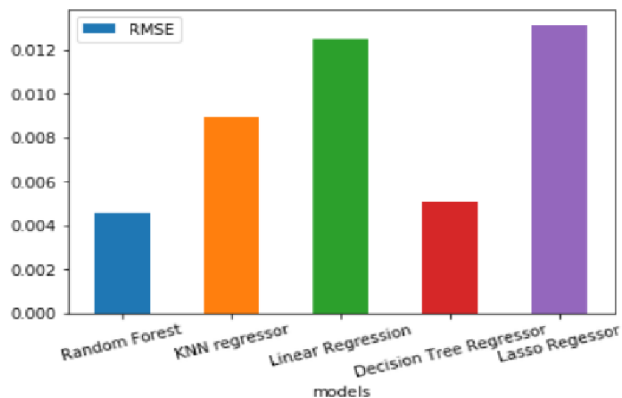
4.2. Graph comparing R2_Score values in various algorithms



4.3. Graph comparing MAE values in various algorithms



4.4 Graph comparing RMSE values in various algorithms



- The above graphs describe the comparison of various error values of fake news detection.
- To find out the best algorithm which gives less error we have compared the five machine algorithms
- From the analysis after the comparison of machine learning algorithms we got decision Tree and Random Forest gave less error and high R2_score compared to other. We use Random Forest algorithm for prediction as it is collection of decision trees.

4.5 OUTPUTS

Fake news classifier

You are reading the news, and might wonder:
Is this fake news? Or a trustful article?

Now you can find out with this fake news classifier.
Just by pasting the content of an article.

Enter the article:

nightmare--a charismatic Christian who is an unapologetic liberal. His desire to stand up for those who have been scared into silence only increased when he survived an abusive three-year marriage. You may know him on Daily Kos as Christian Dem in NC . Follow him on Twitter @DarrellLucus or connect with him on Facebook . Click here to buy Darrell a Mello Yello. Connect"

Submit

Result

It is most likely a trustful article

Probability of 97.59%

Test another article

Fig 4.1 when news is not fake

Fake news classifier

You are reading the news, and might wonder:
Is this fake news? Or a trustful article?

Now you can find out with this fake news classifier.
Just by pasting the content of an article.

Enter the article:

and in so many places, that it is the source of Shakespeare's determination that "what is past, is prelude." Our elites play out our destinies as if they have no free will " no capacity to break with structured ways of seeing. Yet the middle-echelon specialists keep sending their relatively accurate assessments up the ladder of power. Hope springs eternal."

Submit

Result

It is probably fake news, or an opinion piece.

Probability of 83.79%

Test another article

Fig 4.2 When the news is fake

I

CHAPTER 5

CONCLUSION

The objective of this project is to examine the problems and possible significances related with the spread of fake news. We will be working on different fake news data set in which we will apply different machine learning algorithms to train the data and test it to find which news is the real news or which one is the fake news. As the fake news is a problem that is heavily affecting society and our perception of not only the media but also facts and opinions themselves. By using the artificial intelligence and the machine learning, the problem can be solved as we will be able to mine the patterns from the data to maximize well defined objectives.

5.1 Future Work:

- using neural networks so that they may give better results
- Taking an extra parameter cost so that fake news detection becomes easier.
- To optimize the work to implement in Artificial Intelligence environment.

REFERENCES

- [1] W. Jiang, J. Wu, F. Li, G. Wang, and H. Zheng, "Trust evaluation in online social networks using generalized network flow," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 952–963, Mar. 2016.
- [2] M. Alrubaian, M. Al-Qurishi, A. Alamri, M. Al-Rakhami, M. M. Hassan, and G. Fortino, "Credibility in online social networks:A survey," *IEEE Access*, vol. 7, pp. 2828–2855, 2019.
- [3] S. Ranganath, S. Wang, X. Hu, J. Tang, and H. Liu, "Facilitating time critical information seeking in social media," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2197–2209, Oct. 2017.
- [4] Z. Zhang, R. Sun, X. Wang, and C. Zhao, "A situational analytic methodfor user behavior pattern in multimedia social networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 520–528, Dec. 2019
- [5] M. Schudson and B. Zelizer, "Fake news in context," in *Understanding and Addressing the Disinformation Ecosystem*. Philadelphia, PA, USA: Annenberg School for Communication, Apr. 2017, pp. 1–4.
- [6] S. Zaryan, "Truth and trust: How audiences are making sense of fake news," M.S. thesis, Media Commun. Studies, Lund Univ. Publications Student Papers, Stockholm, Sweden, Jun. 2017. [Online]. Available: <https://lup.lub.lu.se/student-papers/search/publication/8906886>
- [7] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, Mar. 2018
- [8] R. Sequeira, A. Gayen, N. Ganguly, S. K. Dandapat, and J. Chandra, "A large-scale study of the Twitter follower network to characterize the spread of prescription drug abuse tweets," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 6, pp. 1232–1244, Dec. 2019.

[

SOURCECODE

```
#import libraries for access and functional purpose
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as s

df_input=pd.read_csv("C:/Users/ADITYA/Desktop/dataworld_set.csv")
df_input.shape
df_input.head()
df_input[News].nunique()
df_input[News].value_counts()
df_input['District_Name'].nunique()
df_input['District_Name'].value_counts()
df_input['Season'].nunique()
df_input['Season'].value_counts()
df_input['Fake_news'].nunique()
df_input['Fake_news'].value_counts()
df_input.info()
df_input.isnull().sum()
df_input.duplicated().sum()
df_input = df_input[df_input['Area'] > 0 ]
df_input = df_input[df_input['Production'] > 0 ]
df_input = df_input.replace(r'^\s*$', np.NaN, regex=True)
df_input = df_input.dropna()
df_input["ProductionPerArea"] = ((df_input["Production"])/(df_input["Area"]))
df_input = df_input.drop(columns=['State_Name','Area','Production'])
News = df_input['Fake_news']
NewsCount = {}
for Fake News in News:
    NewsCount[Fake News ] = NewsCount.get(News, 0)+1
#extract values and keys of dict:NewsCount
labels = list(NewsCount.keys())
```

```

values = list(NewsCount.values())
plt.pie(values, labels=labels, autopct='%1.1f%%')
plt.show()

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

categorical_columns = ['District_Name', 'News', 'Season']
#label encoder dict
labels_dict = {}
#scaling dict
scaling_dict = {}
for column in categorical_columns:
    le = LabelEncoder()
    le.fit(df_input[column])
    df_input[column] = le.transform(df_input[column])
    labels_dict[column] = le.classes_
df_input.describe()
x = df_input['ProductionPerArea']
print(x.describe())

import matplotlib.pyplot as plt
plt.style.use('ggplot')
fig, ax = plt.subplots()
ax.boxplot((x), vert=False, showmeans=True, meanline=True,
           labels=('x'), patch_artist=True,
           medianprops={'linewidth': 2, 'color': 'purple'},
           meanprops={'linewidth': 2, 'color': 'red'})
plt.show()

def deviationTransform(arr):
    d = np.std(arr)
    return [0,d]

def minMaxTransform(arr):
    min = np.min(arr)
    max = np.max(arr)

```

```

    return [min,max-min]
for column in categorical_columns:
    scaling_params = minMaxTransform(np.array(df_input[column]))
    df_input[column] = (df_input[column] -
scaling_params[0])/scaling_params[1]
    scaling_dict[column] = scaling_params
scaling_params = minMaxTransform(np.array(df_input[News]))
scaling_dict[News] = scaling_params
df_input[News] = (df_input[News] - scaling_params[0])/scaling_params[1]
scaling_params =
deviationTransform(np.array(df_input['ProductionPerArea']))
df_input['ProductionPerArea'] = (df_input['ProductionPerArea'] -
scaling_params[0])/scaling_params[1]
scaling_dict['ProductionPerArea'] = scaling_params

p1 = np.percentile(np.array(df_input['ProductionPerArea']), 25)
p2 = np.percentile(np.array(df_input['ProductionPerArea']), 99)
df_input = df_input[df_input['ProductionPerArea'] > p1]
df_input = df_input[df_input['ProductionPerArea'] < p2]
from sklearn.model_selection import train_test_split
df_small = df_input
df_small.columns.name = None
df=df_small
x_train, x_test, y_train, y_test = train_test_split(df.iloc[:,-1], df.iloc[:,-1],
test_size=0.2, random_state=2)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import r2_score
def trained(clf,x_train,x_test,y_train,y_test):
    clf.fit(x_train,y_train)
    y_train_pred = clf.predict(x_train)
    print("MSE " + str(mean_squared_error(y_train, y_train_pred)))

```

```

print("MAE " + str(mean_absolute_error(y_train, y_train_pred)))
print("RMSE " + str(np.sqrt(mean_squared_error(y_train, y_train_pred))))
#print("RMSLE " + str(np.sqrt(mean_squared_log_error(y_test, y_pred))))
R2_score = r2_score(y_train, y_train_pred)
print("R2 Score " + str(R2_score))
y_pred = np.array(y_train_pred)
y_test = np.array(y_train)
data={}
data['x'] = y_test
data['y'] = y_pred
sns.regplot(x="x", y="y", data=data);

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import r2_score
def tested(clf,x_train,x_test,y_train,y_test):
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print("MSE " + str(mean_squared_error(y_test, y_pred)))
    print("MAE " + str(mean_absolute_error(y_test, y_pred)))
    print("RMSE " + str(np.sqrt(mean_squared_error(y_test, y_pred))))
    #print("RMSLE " + str(np.sqrt(mean_squared_log_error(y_test, y_pred))))
    R2_score = r2_score(y_test,y_pred)
    print("R2 Score " + str(R2_score))
    y_pred = np.array(y_pred)
    y_test = np.array(y_test)
    data={}
    data['x'] = y_test
    data['y'] = y_pred
    sns.regplot(x="x", y="y", data=data);

```

Random Forest :

```

from sklearn.ensemble import RandomForestRegressor

```

```

regresser = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
print("\t\t\t random-forest classifier")
trained(regresser,x_train,x_test,y_train,y_test)

```

```

from sklearn.ensemble import RandomForestRegressor
regresser = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
print("\t\t\t random-forest classifier")
tested(regresser,x_train,x_test,y_train,y_test)

```

```

from sklearn.ensemble import RandomForestRegressor
regresser = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
regresser.fit(x_train,y_train)
y_pred = regresser.predict(x_test)
y_pred = np.array(y_pred)
y_test = np.array(y_test)
for i in range(y_test.size):
    if(y_test[i]>0 and y_test[i]<0.001):
        print(i,y_test[i],y_pred[i])
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import r2_score
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
def classify(clf,x_train,x_test,y_train,y_test):
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print(y_pred)
    y_train_pred = clf.predict(x_train)
    metricsList = []
    metricsList.append(mean_squared_error(y_train, y_train_pred))
    metricsList.append(mean_absolute_error(y_test, y_pred))

```

```

metricsList.append(np.sqrt(mean_squared_error(y_test, y_pred)))
metricsList.append(r2_score(y_test,y_pred))
return metricsList

clfMetrics = []
from sklearn.model_selection import train_test_split
df_small = df_input
df_small.columns.name = None
df=df_small

x_train, x_test, y_train, y_test = train_test_split(df.iloc[:, :-1], df.iloc[:, -1],
test_size=0.2, random_state=2)

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
print("\t\t\t random-forest classifier")
clfMetrics.append((classify(regressor,x_train,x_test,y_train,y_test)))
print(clfMetrics)
print(clfMetrics)

```

Html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='css/main.css') }}"
    />
    <title>Fake News Detection</title>
  </head>
  <body>
    <div class = output>
      <h1>Fake News Detection</h1>

```

```

<form>
  <br><label for="stateName">Choose a State:</label>
  <select name="stateName" id="stateName"
onChange="stateSelected()"></select></br>
  <br><label for="districtName">Choose a District:</label>
  <select name="districtName" id="districtName"></select></form></br>
  <br><label for="Fake News ">Choose a Fake News (optional):</label>
  <select name="Fake News " id="Fake News "></select></br>
  <br><label for="season">Choose a Season:</label>
  <select name="season" id="season"></select></br>
  <br><label for="year">Year:</label>
  <input type="text" name="year" id="year" value="2022"></br>
  <button onClick="predict()">Submit</button>
</form>
<div class= results>
<p>Yield in tonnes/hectare ↓⬇⬇</p>
<p id="results">Submit the Form to get Results 🤖 </p>
</div>
</div>
</body>
<script src="{{ url_for('static', filename='js/main.js') }}"></script>
</html>

```

Css

```

html {
  overflow-y: auto;
}
body {
  background-image: url("Fake News .jpeg");
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: 100% 100%;
  color: yellow;
}

```

```

    }
.output {
    font-size: 24px;
    color: yellow;
    border-radius: 1em;
    padding: 1em;
    top: 50%;
    left: 50%;
    margin-right: -50%;
    transform: translate(-50%, -50%);
    position: absolute;
    bottom: 0;
}
table,
tr,
td {
    font-size: 20px;
    font-family: monospace;
    font-weight: bold;
}
h1 {
    font-size: 40px;
    font-family: monospace;
    text-align: center;
}
.results {
    color: rgb(0, 255, 0);
    overflow-y: auto;
    height: 600px;
    width: 500px;
}

```

Javascript:

```

let stateDistrictMap;

fetch(new URL(document.URL) + "/stateDistrictMap")
  .then((res) => res.json())
  .then((data) => {
    stateDistrictMap = data["result"];
    document.getElementById("stateName").innerHTML = List2PicklistValue(
      Object.keys(stateDistrictMap)
    );
  })
  .catch((error) => {
    console.log("error getting states and districts from server :(");
  });

fetch(new URL(document.URL) + "options?param=Season")
  .then((res) => res.json())
  .then((data) => (document.getElementById("season").innerHTML =
    data.result))
  .catch((error) => {
    console.log("error getting seasons from server :(");
  });

fetch(new URL(document.URL) + "options?param=Fake News ")
  .then((res) => res.json())
  .then((data) => (document.getElementById("Fake News ").innerHTML =
    data.result))
  .catch((error) => {
    console.log("error getting News from server :(");
  });

function predict() {
  document.getElementById("results").innerHTML = "Fetching Data ⌂⌂⌂";let
  body = {
    district: document.getElementById("districtName").value,
    season: document.getElementById("season").value,
    year: document.getElementById("year").value,
    Fake News : document.getElementById("Fake News ").value,

```

```

};
if (document.getElementById("stateName").value == "Select") {
    document.getElementById("results").innerHTML = "Select a State";
    return;
}
if (body.district == "Select") {
    document.getElementById("results").innerHTML = "Select a District";
    return;
}
if (body.season == "Select") {
    document.getElementById("results").innerHTML = "Select a Season ";
    return;
}
if (body.year == "") {
    document.getElementById("results").innerHTML = "Fill a Year ";
    return;
}
fetch(new URL(document.URL) + "predict", {
    method: "POST",
    mode: "same-origin",
    cache: "no-cache", // *default, no-cache, reload, force-cache, only-if-
cached
    credentials: "same-origin", // include, *same-origin, omit
    headers: {
        "Content-Type": "application/json",
    },
    redirect: "follow", // manual, *follow, error
    referrerPolicy: "no-referrer", // no-referrer, *no-referrer-when-downgrade,
origin, origin-when-cross-origin, same-origin, strict-origin, strict-origin-when-
cross-origin, unsafe-url
    body: JSON.stringify(body), // body data type must match "Content-Type"
header
})

```

```

.then((res) => {
  if (!res.ok) throw "response status is not ok " + res.status;
  return res.json();
})
.then((data) => {
  let News = data.result;
  let htmlString = "<table><tr><th>Fake News </th><th>Yield</th></tr>";
  for (let i = 0; i < News.length; i++) {
    htmlString +=
      "<tr><td>" +
      News[i][0] +
      "</td><td>" +
      News[i][1].toFixed(2) +
      "</td></tr>";
  }
  htmlString += "</table>";
  document.getElementById("results").innerHTML = htmlString;
  return;
})
.catch((error) => {
  console.log("error while prediction " + error.data);
  document.getElementById("results").innerHTML =
    "Something Went Wrong ";
});
}

document.getElementById("districtName").innerHTML =
"<option value='Select'>Select A State</option>";
function List2PicklistValue(list) {
  result = "<option value='Select'>Select</option>";
  for (idx in list) {
    result += "<option value='" + list[idx] + "'>" + list[idx] + "</option>\n";
  }
}

```

```

    return result;
}
function stateSelected() {
    let state = document.getElementById("stateName").value;
    if (state == "Select") {
        document.getElementById("districtName").innerHTML =
            "<option value='Select'>Select A State</option>";
    } else {
        document.getElementById("districtName").innerHTML =
            List2PicklistValue(
                stateDistrictMap[state]
            );
    }
}
}

```

Python code:

```

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
import os

class model:
    def __init__(self):
        self.regressor = None
        self.params = {}
        self.labels_dict = {}
        self.scaling_dict = {}
        self.df = None

    def deviationTransform(self, arr):
        d = int(np.std(arr))
        return [0, d]

```

```

def minMaxTransform(self, arr):
    min = int(np.min(arr))
    max = int(np.max(arr))
    return [min, max-min]

def getData(self, fileName):
    self.df = pd.read_csv(fileName)

def preProcessData(self):
    if(os.getenv('PY_ENV') != None):
        self.df = self.df.sample(n=1000)
    self.df = self.df[self.df['Area'] > 0]
    self.df = self.df[self.df['Production'] > 0]
    self.df["ProductionPerArea"] = (
        (self.df["Production"])/(self.df["Area"]))
    # replace empty strings with nan
    self.df = self.df.replace(r'^\s*$', np.NaN, regex=True)
    # drop null values
    self.df = self.df.dropna()
    p1 = np.percentile(np.array(self.df['ProductionPerArea']), 25)
    p2 = np.percentile(np.array(self.df['ProductionPerArea']), 99)
    self.df = self.df[self.df['ProductionPerArea'] > p1]
    self.df = self.df[self.df['ProductionPerArea'] < p2]
    self.labels_dict['State_District_Map'] = {}
    for _, row in self.df.iterrows():
        key = row['State_Name']
        value = row['District_Name']
        if key in self.labels_dict['State_District_Map']:
            self.labels_dict['State_District_Map'][key].add(
                row['District_Name'])
        else:
            self.labels_dict['State_District_Map'][key] = set(
                [row['District_Name']])
    for key, value in self.labels_dict['State_District_Map'].items():

```

```

        self.labels_dict['State_District_Map'][key] = list(value)
    self.df = self.df.drop(columns=['State_Name', 'Area', 'Production'])
def encodeAndNormalizeData(self):
    categorical_columns = ['District_Name', 'News', 'Season']
    for column in categorical_columns:
        le = LabelEncoder()
        le.fit(self.df[column])
        self.df[column] = le.transform(self.df[column])
        self.labels_dict[column] = list(le.classes_)
    scaling_params = self.minMaxTransform(np.array(self.df[column]))
    self.df[column] = (self.df[column] -
                        scaling_params[0])/scaling_params[1]
    self.scaling_dict[column] = scaling_params
    scaling_params = self.minMaxTransform(np.array(self.df[News]))
    self.scaling_dict[News] = scaling_params
    self.df[News] = (self.df[News] -
                     scaling_params[0])/scaling_params[1]
    scaling_params = self.deviationTransform(
        np.array(self.df['ProductionPerArea']))
    self.df['ProductionPerArea'] = (
        self.df['ProductionPerArea'] - scaling_params[0])/scaling_params[1]
    self.scaling_dict['ProductionPerArea'] = scaling_params
def classify(self, x_train, x_test, y_train, y_test):
    self.regressor.fit(x_train, y_train)
    y_pred = self.regressor.predict(x_test)
    y_train_pred = self.regressor.predict(x_train)
    print(r2_score(y_train, y_train_pred))
    print(r2_score(y_test, y_pred))
def dumpData(self):
    json_params = {}
    json_params['labels'] = self.labels_dict
    json_params['scaling'] = self.scaling_dict

```

```

self.params = json_params

def predict(self, district, season, year, Fake News ):
    if(not year or year < 2000 or year > 2099):
        return [['Year needs to be between 2000 and 2099', '']]
    district_value = self.params['labels']['District_Name'].index(district)
    district_params = self.params['scaling']['District_Name']
    district_scaled = (
        district_value - district_params[0])/district_params[1]
    season_value = self.params['labels']['Season'].index(season)
    season_params = self.params['scaling']['Season']
    season_scaled = (season_value -
season_params[0])/season_params[1]
    year_params = self.params['scaling']['News']
    year_scaled = (year - year_params[0])/year_params[1]
    Fake News _params = self.params['scaling']['News']
    production_params = self.params['scaling']['ProductionPerArea']
    predictions = []
    News_predicted = []
    if(Fake News == " or Fake News == 'Select'):
        News = self.params['labels']['News']
        for index, _ in enumerate(self.params['labels']['News']):
            Fake News _scaled = (index - Fake News _params[0])/Fake News
_params[1]
            dfPred = pd.DataFrame(
                [[district_scaled, year_scaled, season_scaled, Fake News
_scaled]], columns=['District_Name', News, 'Season', News])
            predictions.append(self.regressor.predict(
                (dfPred)))
            News_predicted.append(
                [News[index],
predictions[index][0]*production_params[1]+production_params[0]])
        return News_predicted, 200
    else:

```

```

        Fake News _value = self.params['labels'][News].index(Fake News )
        Fake News _scaled = (Fake News _value - Fake News
_params[0])/Fake News _params[1]
        dfPred = pd.DataFrame(
            [[district_scaled, year_scaled, season_scaled, Fake News
_scaled]], columns=['District_Name', News, 'Season', News])
        production_pred = self.regressor.predict(dfPred)
        return [[Fake News ,
production_pred[0]*production_params[1]+production_params[0]], 200
def main(self):
    try:
        self.getData('dataworld_set.csv')
        self.preProcessData()
        self.encodeAndNormalizeData()
        x_train, x_test, y_train, y_test = train_test_split(
            self.df.iloc[:, :-1], self.df.iloc[:, -1], test_size=0.33,
random_state=42)
        self.regressor = RandomForestRegressor(
            n_estimators=10, random_state=0)
        print("\t\t\t random-forest classifier")
        self.classify(x_train, x_test, y_train, y_test)
        self.dumpData()
    except Exception as err:
        print("error running model.py main method ", type(err), err)

if __name__ == '__main__':
    mode = model()
    mode.main()

```

Flask:

```
import json
from flask import Flask, render_template, make_response, request
from model import model
import os
app = Flask(__name__)
classParams = {}
port = int(os.environ.get('PORT', 5000))
```

```
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')
@app.route('/stateDistrictMap', methods=['GET'])
def getStateDistrictMap():
    result = clf.params['labels']['State_District_Map']
    return {'result': result}
@app.route('/options')
```

```
def getOptions():
    result = '<option value=\'Select\'>Select</option>'
    for option in clf.params['labels'][request.args.get('param')]:
        result += '<option value=\'{option}\'>{option}</option>'.format(
```

```

        option=option)
    return {'result': result}
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    body = json.loads(request.data)
    response = {}
    try:
        response = clf.predict(
            body['district'], body['season'], int(body['year']), body[News])
        return make_response({"result": response[0]}, response[1])
    except Exception as err:
        print("while predicting error has occurred " + str(err))
        return make_response("error while prediction", 500)
clf = model()
clf.main()
app.run(host='0.0.0.0', port=port, debug=False)

```

PLAGARISM REPORT

FAKE NEWS DETECTION USING MACHINE LEARNING APPROACHES.docx

ORIGINALITY REPORT

3 %	3 %	2 %	1 %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian Institute of Technology, Madras Student Paper	1 %
2	www.springerprofessional.de Internet Source	1 %
3	www.researchgate.net Internet Source	1 %
4	www.scribd.com Internet Source	<1 %
5	scitepress.org Internet Source	<1 %
6	journalofcommunication.ro Internet Source	<1 %
7	www.ijser.in Internet Source	<1 %

PUBLISHED PAPER

1 WELFake: Word Embedding Over Linguistic Features for Fake News Detection

GOVINDAREDDYPALLI CHANDU

Department of Computer Science And Engineering
Sathyabama University Of Science And Technology
Chennai, India
bhargavreddy15072000@gmail.com

Mrs. REFONAA

Department of Computer Science and Engineering
Chennai, India

Abstract— Social media is a popular medium for the dissemination of real-time news all over the world. Easy and quick information proliferation is one of the reasons for its popularity. An extensive number of users with different age groups, gender, and societal beliefs are engaged in social media websites. Despite these favorable aspects, a significant disadvantage comes in the form of fake news, as people usually read and share information without caring about its genuineness. Therefore, it is imperative to research methods for the authentication of news. To address this issue, this article proposes a two-phase benchmark model named WELFake based on word embedding (WE) over linguistic features for fake news detection using machine learning classification. The first phase preprocesses the data set and validates the veracity of news content by using linguistic features. The second phase merges the linguistic feature sets with WE and applies voting classification. To validate its approach, this article also carefully designs a novel WELFake data set with approximately 72 000 articles, which incorporates different data sets to generate an unbiased classification output. Experimental results show that the WELFake model categorizes the news in real and fake with a 96.73% which improves the overall accuracy by 1.31% compared to bidirectional encoder representations from transformer (BERT) and 4.25% compared to convolutional neural network (CNN) models. Our frequency-based and focused analyzing writing patterns model outperforms predictive-based related works implemented using the Word2vec WE method by up to 1.73%.

Index Terms— Bidirectional encoder representations from transformer (BERT), convolutional neural network (CNN), fake news, linguistic feature, machine learning (ML), text classification, voting classifier, word embedding (WE).

I. INTRODUCTION

NOWADAYS people around the world are getting much involved on online social networks regardless of age, community, or sex [1]. Communicating using social networks

is simple, fast, and attractive to share and transfer information. Currently, social network sites like Facebook trailed by Twitter are the market pioneers, facilitating over 1.3 billion clients with a dynamic monthly variation of 300 million users in average [2]. Their collaborations generate Terabytes of information every second [3], [4]. Online social networks are attractive because of the simple and convenient way to access and circulate information with other people. However, the fast scattering of data at a high rate with minimal effort enables the widespread of false information, such as fake news, which are harmful to society and people.

Fake news are low-quality information with purposefully false data, propagated by individuals or bots that deliberately manipulate message for tattle or political plans. Schudson and Zelizer [5] claimed that the term “fake news” originated in previous centuries together with the mass media itself. Nevertheless, this term attracted increased attention after the U.S. presidential elections of 2016, when the propagation of fake news on social media pulled the attention of a larger number of online users than traditional newsreaders. In the last five months before the elections, approximately 7.5 million tweets contained a link to exceptionally one-sided or false news websites. An interesting and worrying aspect is that false and unsubstantiated news from doubtful sources attracts more audiences than credible information [6]. Relevant work on this topic concluded that fake news spread quicker, penetrate further, and have a deeper impact than true news [7]. There are numerous cases where people accept and spread news without checking their correctness certified by sources. By doing this, they become part of a group that deliberately or unintentionally propagates fake news. The intention behind the proliferation of fake news may be manipulation of public views for financial or political benefit, or simply fun. The negative consequences of this phenomenon are, therefore, undeniable, ranging from wrong decision-making to episodes of bullying and violence. Fig. 1(a) and (b) shows two common examples of fake news over social networks.

False information categories are fake news, satire, misinformation, rumor, hoax, disinformation, propaganda, and opinion spam [8]. These categories are not mutually exclusive, but many researchers used them with different storylines. Although there exist a few websites to check the authenticity of the news like PolitiFact [9], The Washington Post Fact Checker [10], FactCheck [11], Snopes [12], TruthOrFic-

Manuscript received October 2, 2020; revised January 26, 2021 and February 25, 2021; accepted March 18, 2021. This work was supported in part by the European Union’s Horizon 2020 Research and Innovation Program through ARTICONF Project under Grant 825134. (Corresponding author: Prateek Agrawal.)

Pawan Kumar Verma is with the Department of Computer Engineering and Applications, GLA University, Mathura 281406, India, and also with School of Computer Science and Engineering, Lovely Professional University, Phagwara 144411, India (e-mail: pawankumar.verma@gla.ac.in).

Prateek Agrawal is with the School of Computer Science and Engineering, Lovely Professional University, Phagwara 144411, India, and also with the Institute of Information Technology, University of Klagenfurt, 9020 Klagenfurt, Austria (e-mail: prateek.agrawal@lpu.co.in).

Ivone Amorim is with MOG Technologies, 4470-605 Moreira, Portugal, and also with the CMUP Mathematical Research Center, University of Porto, 4099-002 Porto, Portugal (e-mail: ivone.amorim@mog-technologies.com).

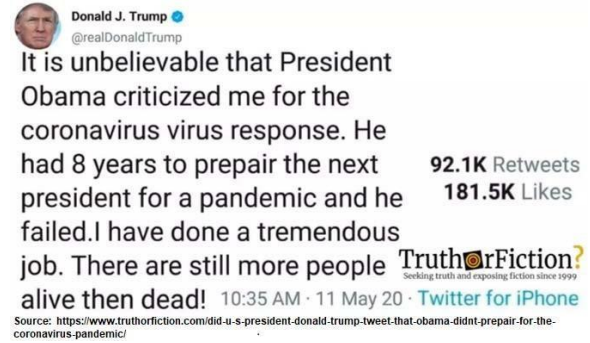
Radu Prodan is with the Institute of Information Technology, University of Klagenfurt, 9020 Klagenfurt, Austria (e-mail: radu@itec.aau.at).

Digital Object Identifier 10.1109/TCSS.2021.3068519

tion [13], FullFact [14], HoaxSlayer [15], Vishvas News [16], unable to spontaneously react to any fake news event [18].
Factly Media & Research [17] yet, these websites are



(a)



(b)

Fig. 1. Fake news examples. (a) Decontextualized news. (b) False news.

As online social networks are major sources of information that can mislead individuals or communities [19], there is a serious need for solutions to verify the authenticity of the content. Many researchers consistently try to develop machine learning (ML) models with different sets of features targeted toward automating the fake news detection process [20], [21] using visual [22] or text-based linguistic approaches. However, the following four questions remain unanswered.

- 1) Which linguistic features are most significant in classifying the news data into real and fake?
- 2) Which word embedding (WE) technique with linguistic features predicts fake news better than other ML methods like convolutional neural networks (CNNs) [21] or bidirectional encoder representations from transformers (BERTs) [23]?
- 3) Which classification method is the most appropriate for fake news detection on available data sets?
- 4) Does ensemble voting classifier improve the fake news detection results?

To answer these questions, we propose a new method called WELFake exclusively focused on text data in three stages.

- 1) Fake news prediction using *linguistic feature* sets (LFS);
- 2) *WE* over LFS for improved fake news detection over a WELFake data set.
- 3) *Comparative analysis* of the linguistic features based results with state-of-the-art CNN and BERT methods.

The WELFake model does not require additional metadata information related to the user or media [24] for the classification of real and fake news. Instead, it aims for a reformation of the state-of-the-art techniques in the detection of fake news over social media websites by using a combined LFS and WE technique. We highlight three contributions of our WELFake model.

A. WELFake Data Set [25]

We designed a larger WELFake data set to prevent overfitting of classifiers and enable better ML training. For this purpose, we merged four popular news data sets (i.e., Kaggle, McIntire, Reuters, and BuzzFeed Political) and prepared a more generic data set of 72 134 news articles with 35028 real and 37 106 fake news.

B. WELFake Fake News Detection Model

We proposed a novel WELFake model for fake news detection in two steps.

- 1) collection of various linguistic features from state-of-the-art methods and identification of a subset that performs well on the larger WELFake data set, and
- 2) ensemble learning on WE features using various ML methods.

C. WELFake Model Generalization and Validation

We applied an adversarial approach to evaluate the model generalization and effectiveness by training and testing on separate data sets.

Experimental results on the WELFake data set revealed that our model achieved a fake news classification accuracy of up to 96.73%, which improves the state of the art by 4.25% over CNN and by 1.31% over BERT methods.

The article has eight sections. Section II gives a background overview on text classification methods. Section III highlights the related work. Section IV describes the proposed methodology and the WELFake model, followed by the resulting algorithm in Section V. Section VI describes the implementation of CNN and BERT state-of-the-art methods for fake news classification used for direct comparison. Section VII provides evaluation results and further related work comparison. Section VIII concludes the article and highlights the future work.

II. TEXT CLASSIFICATION BACKGROUND

This section discusses several ML [26] methods, including CNN and BERT for text classification.

A. ML Classification Methods

We review in this section a few ML methods [26] used for fake news classification in the WELFake model.

1) *Naive Bayes*: This is a supervised learning algorithm based on Bayes' theorem that gives fast predictions with better accuracy in the domain of sentiment analysis, spam filtration, and text classification?

2) *Support Vector Machine*: This is a supervised learning algorithm that works for both classification and regression problems. The algorithm finds the best line for set separation and predicts the correct set for new data values.

3) *Decision Tree*: This is a supervised learning algorithm that classifies the data for both categorical and continuous dependent variables. This classifier uses tree structures to solve a problem by distributing complete data sets into homogeneous ones. Internal nodes, branches and leaf nodes in this tree structure represent the data set, the decision rules and the outcome. There are two attribute selection measures for the best attribute node: information gain and Gini index.

4) *Random Forest*: This is a supervised learning algorithm based on ensemble learning that ensembles several decision trees (DTs) into a random forest (RF) and calculates the average results. The large number of trees in the RF may increase the model accuracy.

5) *K-Nearest Neighbor*: This is majorly useful for classification problems based on feature similarity. The algorithm can use any integer value for K based on the problem statement and statistics, and employs the Euclidean, Manhattan, or Hamming metric for calculating the distance between data.

6) *Boosting*: This connects all base learners sequentially. Initially, it passes a few records to the first base learner (BL_1) (of any model) for training, evaluates all the records on BL_1 , and passes the incorrectly classified ones to the second learner (BL_2) for training. BL_2 tests all the records and passes the incorrectly classified ones to the next learner BL_3 . This process continues until a specified number of base learners.

7) *Bagging*: This is known as bootstrap aggregation, this is an ensemble technique that uses multiple base learners and provides different subsets of the original data set to each model for training (bootstrapping). The testing process decides the output based on the majority votes from the different models (aggregation). Apart from different sample sets, one can train the models with different subsets to reduce over-fitting.

B. Text Classification Methods

This section reviews the state-of-the-art CNN and BERT classification methods used in our experimental evaluation.

1) *CNN* [27]: This is each sentence into words (called tokens) and converts them into vectors using context-based WE methods like GloVe, Word2vec, and FastText. These vectors join together to form a $m \times n$ matrix for a given sentence, where m is the embedding dimension and n is the number of tokens present in the vocabulary. Next, it applies the filters on the 1-D convolution (Conv1D) layer. This convolution kernel works from top to bottom (single directional) because of the same width of filters. The Conv1D later combines its outcomes and applies pooling (maximum, average, and global) that passes the data to the final fully connected layer. This takes care of the output generated by the pool layer and produces the classification decision depending on the loads appointed to each other inside the text.

2) *BERT* [28]: This is a popular pretrained model for text processing developed by Google, which gives a better sense of the language context compared to unidirectional models in two phases.

1) Bidirectional training of an input and output transformer representation model, comprising:

a) an encoder reading the text as input, and

b) a decoder generating the output based on the task; and

2) Popular attention mechanism with a neural network (NN) implementing important features only [29].

BERT takes input text and preprocesses it using tokenization, lemmatization, stopword removal, and text lowering operations. Second, it passes the preprocessed text to the encoding phase where additional token, segment, and positional embedding processes take place. Third, it converts the input into default 768 long embedding vectors and passes them through the encoder layers. Finally, the more accurate information of each token available at the last encoding layer passes to the dense layer for text categorization.

III. STATE-OF-THE-ART SURVEY

This section focuses on reviewing various important fake news detection methods.

A. Clickbaits

Clickbaits a special type of fake content that contains linguistic headlines to attract the readers but do not fulfill their promises. Chen *et al.* [30] scrutinized potential techniques for programed discovery of clickbaits by combining syntax and semantics under textual cues, as well as images and newsreader behavior under nontextual cues. They surveyed the importance of these cues for the identification of fake news but failed to implement a prototype of their methodology. Bourgonje *et al.* [31] proposed a model for clickbait detection that checks the relevance of headlines for article bodies. They used the data set released by the coordinators of the first fake news challenge on stance detection and achieved a significant accuracy of 89.59% using a logistic regression classifier. Rashkin *et al.* [32] compared the language of real news with satire, hoaxes, and propaganda, and discovered the features of fake text and other online sources for the fact-checking.

B. News Credibility

Alrubaian *et al.* [33] analyzed the credibility of 489 330 Twitter accounts using text (sentiment analysis), the user (gender, number of followers), and message (URL, hashtag, number of replies) specific features. They implemented RF, Naive Bayes (NB), DT, and feature-rank NB algorithms with a ten-fold cross-validation. Castillo *et al.* [34] detected fake news based on credibility using four feature sets: text-specific (e.g., number of characters, words, question marks, punctuation, and sentiment analysis), user-specific (e.g., verified accounts, number of followers, number of tweets, and account creation time), message-specific (e.g., with URL, retweeted status) and propagation (e.g., root node degree and propagation tree depth). These features applied on a Twitter data set obtained precision and recall in the range between 70% and 80%. Benjamin *et al.* [35] classified the news as real or fake based on stylistic (e.g., syntax, text style, and grammar), complexity (e.g., article title), and psycholinguistic features. Experimental results on three data sets revealed better results with large amounts of

data and more in-depth features. Kaliyar *et al.* [36] proposed a deep convolutional neural network for fake news detection network (FNDNet) model and achieved a 98.36% accuracy on the Kaggle data set. However, they did not show results on generalized text [37]. Shu *et al.* [38] discussed the worthiness of real or fake news detected through comments on particular items and achieved 90.4%, respectively, 80.80% accuracy on the PolitiFact and GossipCop data sets. Zellers *et al.* [39] proposed a text generation model named GROVER that uses adversaries to spread more trustworthy disinformation than humans. They investigated several propagated threats and showed that GROVER can be an effective discriminator that outperforms BERT in detecting the generated false news.

C. Linguistic Features

The utilization of linguistic features for fake news detection has been popular since the mid-2000s. Burgoon *et al.* [40] used 16 linguistic features categorized in four classes, which achieved an accuracy of 60.72% using a DT algorithm with 15-fold cross-validation. Vicario *et al.* [41] used different features like text (e.g., number of characters, words, sentences, question marks, and negations), user-specific, and message-specific (e.g., number of replies, likes) to identify hoaxes and fake news on social media using linear regression, logistic regression, support vector machine (SVM), K-nearest neighbor (KNN), and NNs. The validation on an Italian Facebook data set with new features achieved an accuracy of 91% on the linear regression classification algorithm. Pérez-Rosas *et al.* [42] used major linguistic features (e.g., n-grams, punctuation, psycho-linguistic, readability, and syntax) and achieved an accuracy of 76% on two novel data sets covering seven domains. Buntain and Golbeck [43] attempted to classify real or fake Twitter threads in four categories using 45 features: structural (e.g., length, the number of tweets), content (e.g., polarity, subjectivity), temporal, and user (e.g., age, followers, authenticity). They used the CREDBANK [44] and PHEME [45] data sets for training and the BuzzFeed data set for testing with an accuracy of 65.29%. Newman *et al.* [46] applied logistic regression to evaluate 29 features and achieved an accuracy between 52% and 67%. Similarly, Zhou *et al.* [47] used 20 features of nine categories for fake news detection. Ahmed *et al.* [48] used an unclear private version of the continuously updated Kaggle data set for fake news detection and achieved an accuracy of 92%. Shu *et al.* [49] used Politifact and BuzzFeed data sets in their analysis, which are hard to generalize due to their very small size. Gravanis *et al.* [37] compared their model with [48], [49] by combining some articles from four data sets (Kaggle-EXT, McIntire, BuzzFeed, and Politifact) in a new UNBiased data set with 3004 articles and achieved the highest accuracy of 95% using SVM. Table I shows a comparative study of five linguistic feature categories.

- 1) *Readability index* quantifies the text's complexity (i.e., reading difficulty) based on word length, number of syllables, and sentence length.
- 2) *Psycho-linguistic* features describe emotions, behaviors, persona, and mindset.
- 3) *Stylistic* features explain the style of a sentence.
- 4) *User credibility* features describe user information.

TABLE I
STATE-OF-THE-ART COMPARISON OF LINGUISTIC FEATURES

<i>Linguistic feature</i>	[35]	[41]	[48]	[49]	[37]	WELFake
Readability index	✓	✗	✗	✗	✓	✓
Psycho-linguistic	✓	✗	✗	✓	✓	✓
Stylistic	✓	✓	✗	✗	✓	✓
User credibility	✗	✓	✗	✓	✗	✗
Quantity	✗	✗	✗	✗	✓	✓

TABLE II
FAKE NEWS DETECTION DATA SETS

<i>Dataset</i>	<i>Articles</i>	<i>Categories</i>
Benjamin Political News [35]	225 stories	Real, fake, satire
Burfoot Satire News [50]	4,233 news	Real, satire
BuzzFeed News [35]	101 news	Real, fake
CREDBANK [44]	60 million tweets	Real, fake
Fake News Challenge [51]	50,000 articles	Agree, disagree, discuss, unrelated
FakeNewsNet [49]	422 news	Real, fake
LIAR [52]	12,800 news	True (mostly, half, barely), false, pants-fire
Reuters [48]	44,898 articles	Real, fake
McIntire [53]	6,335 news	Real, fake
Kaggle [54]	38,729 news	Real, fake

- 5) *Quantity* features explain sentence information such as the number of words and number of sentences.

D. Open Data Sets

From the large number of data sets available for the study of fake news [55]–[61], we highlight some popular data sets in Table II.

- 1) *Benjamin Political News* is a very small data set with 75 stories each for real, fake, and satire categories.
- 2) *Burfoot Satire News* is an unbalanced data set with real and satire categories.
- 3) *BuzzFeed News* is a very small data set of 101 news.
- 4) *CREDBANK* consists of incomplete news articles in real and fake events categories.
- 5) *Fake News Challenge* focuses on individual claims among three categories: disagree, discuss and unrelated.
- 6) *FakeNewsNet* consists of only 422 news articles with incomplete classification in real and fake categories.
- 7) *LIAR* contains various hard-to-classify social media posts and speeches due to the lack of verification sources or knowledge bases [62].
- 8) *Reuters* consists of real and fake news articles from a single source that increases the chances of biased data.
- 9) *McIntire* consists of real and fake news categories without authentic confirmation from any individual.
- 10) *Kaggle* consists of real and fake news data without source information.

E. Summary

No single method guarantees the best solution for all data sets. The state-of-the-art approach of Gravanis *et al.* [37] used 57 linguistic features and embed them with a word-to-vector embedding method on UNBiased data set of less than 4000 articles only. They used all features in a single LFS

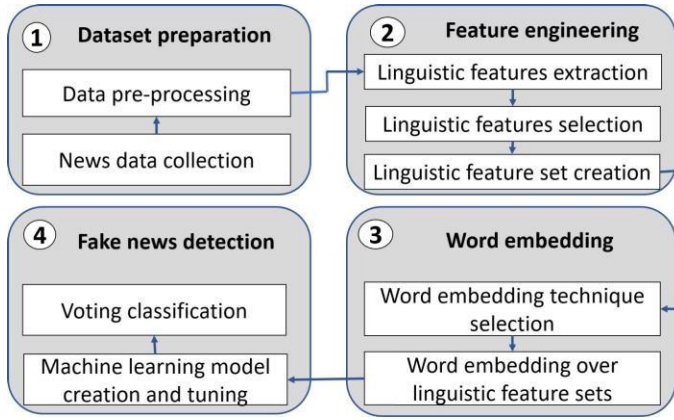


Fig. 2. WELFake model overview.

and achieves up to 95% accuracy. WELFake improves on this method using a novel method based on four stages that:

- 1) creates a larger data set with improved generalization;
- 2) identifies the most significant twenty linguistic features and creates three unique LFS based on categories;
- 3) applies two WE methods to train various ML models; and
- 4) generates the final prediction using a two-stage voting classification.

IV. WELFAKE MODEL

This section outlines the WELFake model for fake news detection divided into four phases, shown in Fig. 2.

- 1) *Data set preparation* involves the collection and pre-processing of the data in a proper format, as a fundamental task of any ML model;
- 2) *Feature engineering* involves linguistic feature extraction and selection;
- 3) *WE* identifies the most appropriate technique connected with the LFS;
- 4) *Fake news detection* tunes the model parameters and applies a hard voting classifier for better accuracy.

A. Data Set Preparation

1) *News Data Collection*: This is essential for a balanced and unbiased data set and the key to providing high quality training data and delivering good results. Although there exist an important number of open data sets for the study of fake news (see Section III and Table II), the literature showed their serious limitations in terms of size, category, or bias. After a careful study, we prepared a more comprehensive *WELFake data set* that combines four data sets, Kaggle, McIntire, Reuters, and BuzzFeed, for two reasons. First, they have a similar structure with two categories (i.e., real and fake news). Second, combining the data sets reduces the limitations and the bias of each individual data set. Table III shows the WELFake open data set with 72 134 news articles classified as 35 028 real and 37 106 fake news [25]. The data set contains three columns (i.e., title, text, label) with a binary label for fake and real news. Table IV summarizes the balanced fake

TABLE III
WELFAKE DATA SET

<i>Dataset</i>	<i>Real news</i>	<i>Fake news</i>
Kaggle	10387	10413
McIntire	3171	3164
Reuters	21417	23481
BuzzFeed Political	53	48
WELFake dataset	35,028	37,106

TABLE IV
FAKE VERSUS REAL NEWS DISTRIBUTION IN WELFAKE

<i>Category</i>	<i>Real news [%]</i>	<i>Fake news [%]</i>
Short sentences	60.9	39.1
Readability index	51.7	48.3
Subjectivity	45.4	54.6
Number of articles	53.9	46.1
WELFake dataset	48.55%	51.45%

and real news distribution in the WELFake data set across all four feature categories.

a) *Number of short sentences* (below ten words) representing real news is greater than those representing fake news.

b) *Text readability* of fake news is poorer than the readability of real news.

c) *Subjectivity* of fake news articles is larger than for real news articles.

d) *Number of articles* representing real news is larger than those representing fake news.

2) *Data Preprocessing*: This solves different problems in the collected data, like typographic errors, unstructured data format, and other limitations, using several methods, depending on the data set and objectives.

a) *Missing data* handles undefined (NaN) and blank values (NULL) present in the data set, which hinder the feature engineering process. Since deleting the data entries containing missing values may cause the loss of important information [63], we performed a missing value imputation process that estimates missing values and then analyzes the complete data set as if these values were the actual observed ones.

b) *Inconsistent data* deviates from other data points because of mistakes during data collection. We employed several visualization techniques and mathematical functions for outlier identification and correction, like box plot, scatter plot, Z-score, and inter-quartile range (IQR) score.

c) *Duplicate data* or deduplication removes redundancy that can lead to biased results, which may occur when the same person collected the data.

d) *Irrelevant data* removes stop words (and other noise) that make the sentence grammatically complete, but do not have semantic significance in news classification operations. Removing stop words and keeping relevant tokens only significantly increases the model performance.

e) *Stemming* converts the text into its root word by applying the Porter–Stemmer algorithm on text features for accuracy improvement. In case it cannot recognize the root word, it generates the canonical form of a corresponding word.

B. Feature Engineering

1) *Linguistic Features Extraction*: This is the process of conversion from raw text to the data provided to the ML

algorithm. Feature extraction aims to create a feature set that summarizes the information of the original data set, which speeds up the model training and improves the data visualization and learning accuracy. We extracted in WELFake 87 text-based linguistic features from state-of-the-art works, falling into two syntactic (i.e., writing pattern, quantity) and semantic (i.e., grammar, psycho-linguistics) categories.

a) *Writing pattern* emphasizes the writing style of the text by accessing the sentence types, use of determinants, special characters, and modifiers.

b) *Grammar* focuses on the text readability index and emphasizes the word structure, average syllables per word, easy word use ratio in a word list, and sentence complexity.

c) *Psycho-linguistic* estimates the text sentiment and information opinion based on semantic and subjectivity.

d) *Quantity* identifies speech information parts by counting the verbs, adjectives, adverbs, syllables, and words, as well as the rate of adjectives, adverbs, and words per sentence.

2) *Linguistic Features Selection*: This is the process of choosing important features for data classification, which diminishes the quantity of input features, decreases the computational expense and improves the accuracy of the ML model. For this purpose, we calculated the Pearson's correlation coefficient of each feature with the other features within the same category and discarded those with a correlation coefficient higher than 0.7, which indicates a strong positive linear relationship [23] between the two features. A learning model with fewer features is simpler and more precise according to Occam's razor principle and the minimum description length concept [62]. We executed this recurrent process until removing all the least significant features. Table V shows the remaining 20 most significant features of WELFake grouped based on the four categories.

3) *LFS Creation*: This groups the selected 20 linguistic features in different sets to enable multiple WE methods for unbiased model training. We need an odd number of input sets to obtain a clear result upon subsequent voting classification. We consequently created a minimum of three distinct LFS based on four categories, as shown in Table V.

a) *Readability index* defines the sentence structure complexity of any text. We identify the level or grade of the text writer based on the readability index, which helps in classifying the news as real or fake. For this reason, we uniformly distributed all three readability index features among the three LFS and assigned one feature to each set.

b) *Psycho-linguistic* features play an important role in detection of fake news, as explained in Section III. We, therefore, used all three of them in all the three LFS.

c) *Quantity* features also participate in news classification, so we evenly distributed them across the three LFS.

d) *Writing pattern* contains five features and, therefore, we evenly distributed three features to each LFS.

C. WE

1) *WE Selection*: This identifies the most appropriate technique for converting plain text into a numeric value, as ML methods cannot directly process plain text. We observed two popular WE categories in the literature: 1) *content-based*, such

TABLE V
WELFAKE LFS

Category	Feature	LFS1	LFS2	LFS3
Writing pattern	Number of special characters	✓	✗	✓
	Number of determinants	✗	✓	✓
	Number of capital letters	✗	✗	✓
	Number of short sentences	✓	✓	✗
	Number of long sentences	✓	✓	✗
Grammar (readability index)	Gunning log grade readability index	✓	✗	✗
	SMOG readability index	✗	✓	✗
	Automatic readability index	✗	✗	✓
Psycho-linguistics	Text polarity	✓	✓	✓
	Title similarity	✓	✓	✓
	Subjectivity	✓	✓	✓
Quantity	Number of syllables	✗	✗	✓
	Number of words	✗	✓	✗
	Rate of adjectives and adverbs	✓	✗	✗
	Words per sentence	✗	✗	✓
	Number of articles	✓	✗	✗
	Number of verbs	✗	✓	✗
	Number of sentences	✗	✓	✗
	Number of adjectives	✓	✗	✗
	Number of adverbs	✗	✗	✓

as term frequency-inverse document frequency (TF-IDF) and count vectorizer (CV), that focus on previous knowledge and 2) *context-based*, such as Word2Vec, GloVe, and FastText, that focus on writing text patterns. As fake news writers (fakesters) tend to repeat similar words, we selected the content-based WE that focuses on writing patterns rather than context [64].

a) *CV*: Also called one-hot encoding, CV converts text document in a histogram vector, where each element represents the number of appearances of the word in the document. The vector length depends on the number of unique words in the corpus.

b) *TF-IDF*: This is the advanced version of CV, which shows the importance of a term (representing a word) in a corpus alongside its occurrence in the document. *TF-IDF* is the multiplication of the term frequency $tf(t, d)$ that computes the occurrence of a term t in a document d and the inverse document frequency $idf(t, D)$ that computes the importance of that term t in a corpus of documents D

$$TF-IDF(t, d, D) = \frac{tf(t, d)}{f_d(t)} \cdot idf(t, D)$$

$$tf(t, d) = \frac{\sum_{d \in D} f_d(t)}{\max_{w \in d} f_d(w)}$$

$$idf(t, D) = \ln \frac{|D|}{|\{d \in D : t \in d\}|}$$

where $f_d(t)$ represents the number of occurrences of the term t in the document d , $|D|$ is the number of documents in the corpus D , and $|\{d \in D : t \in d\}|$ is the number of documents containing the term t .

2) *WE Over LFS*: This improves the output prediction, as predefined features do not always accurately predict and need additional training methods. For this purpose, we combined the WE with LFS. We applied the TF-IDF and CV WE techniques on the three LFS and found that CV gives better results. We achieved maximum of 95.61% accuracy using SVM on CV, while TF-IDF gave maximum accuracy of 95.12% using bagging. Thus, we selected CV and combined it with LFSs for further accuracy analysis of various models in Section VII-B.

TABLE VI
HYPERPARAMETERS FOR ML MODELS TUNING

Model	Iterations	Hyper-parameter	Value range	Value
NB	4	Smoothing	[0,	true
		Fit prior	{ true, false }	
SVM	9	Kernel	{ linear, poly, rbf, sigmoid, precomputed }	linear
		Regularization	{ float }	
		Kernel coefficient	$x x \in \{\text{scale, auto}\}$	
		Degree	$\mathbb{Z}$	
KNN	5	Number of neighbors	$\mathbb{Z}$	2
		Weights	{ uniform, distance, callable }	
DT	8	Maximum number of features	$x x \in \{\text{auto, sqrt, log 2}\}$	gini
		Cost complexity pruning	{ gini, entropy }	
		Number of estimators	{ object, none }	
		Bootstrap	$\mathbb{Z}$	
Bagging				true
AdaBoost	5	Number of estimators	$\mathbb{Z}$	50
		Learning rate	{ true, false }	

D. Fake News Detection

1) *ML Model Creation and Tuning*: This passes the LFS with WE through six ML methods: SVM, NB, KNN, DT, Bagging, and AdaBoost. For this purpose, we experimented with each ML model on random samples of the WELFake data set with four training-testing data combinations: 60%–40%, 70%–30%, 80%–20%, and 90%–10%. To improve the accuracy, we performed a manual tuning of the six different models using the hyperparameters displayed in Table VI. We sequentially explored different hyperparameter value combinations from the given possible value ranges and tuned them until we obtained a state-of-the-art accuracy of at least 96%. We evaluated the performance of each ML model on different training and testing data distributions as explained in Section VII-B and found out that a 70%–30% data distribution gives better accuracy for all six ML methods.

2) *Voting Classification*: This process uses ensemble learning to collect predictive outputs from various models and generates an output that minimizes the error and the overfitting. There are in general two voting classifier approaches: soft voting based on probability and hard voting based on maximum votes. Since fake news detection is a binary classification problem, we use *hard voting* that predicts a target variable Y based on the maximum votes mode given by different models M_i to a class:

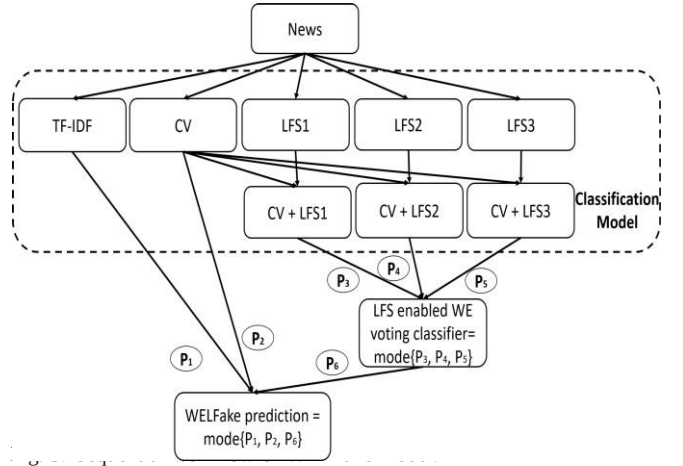
$$Y = \text{mode}\{M_1(X), M_2(X), \dots, M_n(X)\}$$

where X is predictor or input variable.

V. WELFAKE FAKE NEWS DETECTION ALGORITHM

Fig. 3 shows the WELFake binary news classification method (as real or fake) using TF-IDF and CV, three linguistic feature sets (LFS1, LFS2 and LFS3) and the hard voting classifier. The WELFake workflow consists of four steps:

- 1) Apply TF-IDF and CV on entire data set, store the results in P_1 and P_2 , and decide the better WE based on accuracy.



- 2) Apply CV (better performing method) on three well-defined LFS and store the results in P_3 , P_4 , and P_5 .
- 3) Apply WE hard voting classifier on P_3 , P_4 , and P_5 , and generate the prediction P_6 as output.
- 4) Combine P_1 , P_2 , and P_6 using the hard voting classifier and generate the final prediction output.

Algorithm 1 explains the steps of the proposed WELFake model organized in four phases, as explained in Fig. 2.

- 1) *Data Set Preparation*: This the WELFake data collection in line 1 and data set preprocessing in line 2.
- 2) *Feature Engineering*: Extracts linguistic features of the data set in line 3 and applies Pearson's coefficient to select the significant linguistic features in line 4. Line 5 creates an odd number of LFS to perform voting.
- 3) *WE*: (i.e., CV and TF-IDF) This applies on the entire data set in lines 6 and 7. We select in line 8 the best method to combine with the various LFS created in line 5. Lines 9 and 10 combine the LFS with the best WE method.
- 4) *ML Model Tuning and Voting Classifier*: We train in line 11 the data sets from line 5 on various ML classification models and select the best results from each set. Line 12 applies the voting classifier on the results achieved on the different LFS using the best ML classification model and generates the hard voting output. Line 13 applies again the hard voting classifier (line 12), CV (line 6), and TF-IDF (line 7) and returns the final news classification prediction.

VI. CNN AND BERT IMPLEMENTATIONS ON WELFAKE DATA SET

This section describes the CNN and BERT implementation on the WELFake data set. We performed a hyperparameter tuning to improve their performance, summarized in Table VII.

A. CNN-Based Fake News Detection

After preprocessing the text as presented in Section IV-A2, we pass it to our CNN implementation consisting of several layers shown in Fig. 4.

Algorithm 1: WELFake Fake News Detection Algorithm

Data: Kaggle, McIntire, Reuters, BuzzFeed
Result: WELFake Model for news classification

```

// Phase 1: data set preparation
1 WELFake_data set ← collection(Kaggle, McIntire, Reuters, BuzzFeed) // News collection
2 WELFake_data set ← preprocess(WELFake_data set) // Data set pre_processing
// Phase 2: feature engineering
3 LF ← extract(WELFake_data set) // Linguistic feature extraction
4 LF ← selection(LF) // Feature selection using Pearson's correlation
5 LFS ← split(LF) // Split linguistic features in odd sets
// Phase 3: Word embedding
6 CV ← cv(WELFake_data set) // Apply CV technique on data set
7 TFIDF ← tfidf(WELFake_data set) // Apply TFIDF technique on data set
8 Best ← select(CV, TFIDF) // Select best embedding technique with data set
9 foreach LFSi ∈ LFS do
10   CLFSi ← combine(CLFSi, Best)
// Phase 4: ML model tuning and voting classifier
11 Model ← bestModel(SVM (CLFS), DT (CLFS), NB (CLFS), Bagging (CLFS), AdaBoost (CLFS), KNN (CLFS))
// Selection of best model for linguistic feature set
12 vote_hard ← votingClassifier(Model (CLFSi ));
13 return votingClassifier (TFIDF, CV, vote_hard)

```

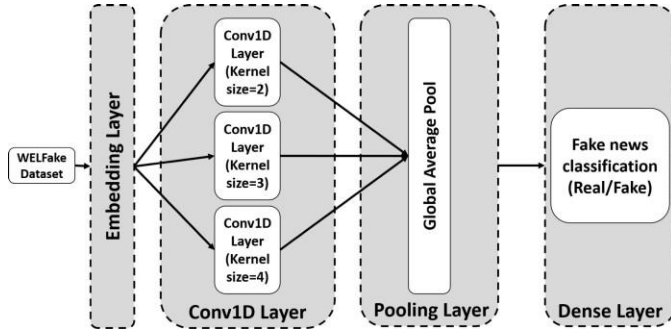


Fig. 4. CNN text processing on WELFake data set.

1) *Embedding Layer*: This converts the input text into WE, where every word in the vocabulary transforms into a high-dimensional space vector for next layer execution. We used the GloVe technique for the conversion of text into a vector.

2) *Convolutional Layer*: This layer the WE from the previous layer as input and specifies the number of Conv1D layers and the kernel size k , representing the number of words in a sentence during one filter slide. We used three Conv1D layers having kernels of size two, three, and four with 32 filters each (totalling 96 filters in the Conv1D layers). We obtained 32 feature maps from each kernel for further execution and calculated the filter height as $m-k+1$, where m is the maximum sequence length in the data set.

3) *Pooling Layer*: The pooling layer reduces the number of outputs generated at the previous layer. After applying a global average pooling, the size of the vector is the same as the number of filters. This layer also merges the converted vectors of size $f \times x$, where f is the number of filters and x is the number of kernels.

4) *Dense Layer*: This layer takes the output from the pooling layer and passes it on to two consecutive dense layers

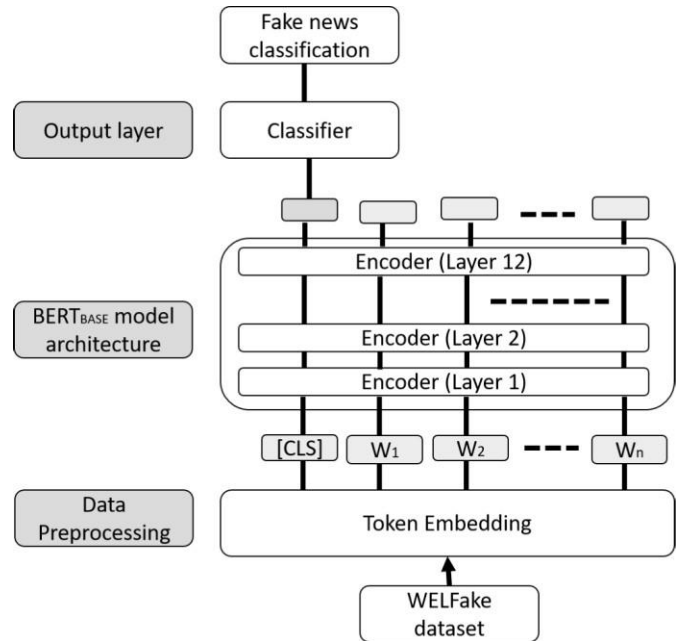


Fig. 5. BERT text processing on WELFake data set.

for final prediction using a Sigmoid activation function with values in the $[0, 1]$ interval, which helps to assess the CNN prediction confidence.

B. BERT-Based Fake News Detection

We performed fine-tuning of the pretrained BERT model for fake news classification in several steps, shown in Fig. 5.

1) *Data Preprocessing*: performs the operations described in Section IV-A2 on the raw text and adds a classification [CLS] token at the start of the first sentence.

TABLE VII
CNN AND BERT MODEL PARAMETERS

Model	Parameter	Value
CNN	Number of Conv1D layers	3
	Number of filters	32
	Kernel Size	2,3,4
	Number of global avgpool layer	3
	Activation function	Sigmoid function
	Optimizer	Adam
BERT	Loss function	Binary-crossentropy
	Number of dense layers	5
	Number of dropout layer	3
	Activation function	Sigmoid function

TABLE VIII
FAKE NEWS PREDICTION PARAMETERS

Evaluation parameter	Predictive value	Actual value
True positive (TP)	Yes	Yes
True negative (TN)	No	No
False positive (FP)	Yes	No
False negative (FN)	No	Yes

2) *BERT Model Architecture*: uses the BERT_{BASE} model with 12 encoder layers. Each encoder layer performs a self-attention mechanism and passes the output to a feed-forward network. The output of each layer is a vector of default size 768. We applied at the 12th layer the [CLS] token to store the useful information for classification.

3) *Output Layer*: reads the vector stored in the [CLS] token. We used a Sigmoid activation function and implemented a NN with five dense layers and three dropout layers to classify the text news. The Sigmoid function produces the results in the range of [0, 1], which identifies confidence classes such that the class with a larger value wins the prediction.

VII. EXPERIMENTAL EVALUATION

We implemented WELFake in Python 3.6 on a Jupyter notebook. We run the experiments on a computer equipped with a ninth-generation i7 processor and 16 GB of memory.

A. Evaluation Metrics

We define four evaluation parameters, true-positive (TP), true-negative (TN), false-positive (FP), and false-negative (FN), based on the relation between the predictive news classification and the actual one, displayed in Table VIII. Based on these parameters, we evaluated the WELFake model on four performance metrics.

1) *Accuracy*: This is the ratio between the number of correct predictions and the total number of predictions

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

2) *Precision*: This measures the positive predicted value, as the ratio between the number of correct positive predictions to the total number of positive predictions

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

3) *Recall*: R measures the sensitivity of the model as the ratio between the number of correct positive predictions to the total number of correctly predicted results

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

4) *F1-Score*: $F1$ measures the testing accuracy of the model as the harmonic mean of the precision and the recall

$$F1_score = \frac{2}{\text{Recall}^{-1} + \text{Precision}^{-1}}.$$

B. Comparative ML Classification

We analyze the accuracy of the WELFake model on the six ML classifiers in Table IX using four scenarios.

1) *LFS*: This classifies the news on the three LFS individually using the six ML classifiers. The accuracy of all three sets ranges between 77.3% and 85.6%. LFS1 gives the best accuracy of 83.4% using Bagging and the worst accuracy of 79.2% using NB. LFS2 produces the highest accuracy of 84.2% using Bagging and the lowest accuracy of 77.3% using NB. Similarly, LFS3 achieves the highest 85.6% accuracy using SVM and the lowest accuracy of 79.8% using NB. Bagging and SVM performed the best among all six ML classifiers followed by AdaBoost, DT, KNN, and NB.

2) *WE*: This applies CV and TF-IDF on the WELFake data set and classifies the news to predict P_1 and P_2 , as illustrated in Fig. 3. We observed that CV achieved in general a better accuracy than TF-IDF. CV performed the best on SVM with a 95.61% accuracy, followed by Bagging, AdaBoost, NB, DT, and KNN. TF-IDF achieved an accuracy of 95.12% using Bagging, followed by SVM, AdaBoost, NB, KNN, and DT.

3) *LFS-Enabled WE*: This combines CV with the three LFS to predict P_3 , P_4 , and P_5 and applies voting classifier to obtain P_6 . We achieved a maximum accuracy of up to 96.1% using SVM and a minimum accuracy of up to 89.6% using DT. Overall, SVM performed the best followed by Bagging, AdaBoost, NB, KNN, and DT.

4) *WELFake Prediction*: This predicts the final classification output by applying voting classifier across the LFS-enabled WE classification (P_6), TF-IDF (P_1) and CV (P_2) predictions. SVM achieved the maximum accuracy of 96.73% followed by AdaBoost, Bagging, NB, KNN, and DT.

5) *Comparison*: Table X shows the performance among several training and testing dataset splits and Table XI compares the overall performance of the WELFake model in terms of accuracy, precision, recall, and F1-score, introduced in Section VII-A. From all six ML models, WELFake produces the best results using SVM with a maximum accuracy of 96.73%, and a minimum accuracy of 89.92% using DT. Similarly, SVM achieved the highest precision (94.6%), recall (98.61%), and F-1 score (96.56%), while DT scored the lowest precision (86.1%) and F1-score (89.24%). KNN achieved the worst recall of 90.55% only. From these results, we conclude that the proposed WELFake model achieved the best performance on the WELFake data set (on all evaluation metrics) using SVM followed by AdaBoost, Bagging, NB, KNN, and DT.

TABLE IX
ACCURACY ANALYSIS OF WELFAKE MODEL

Scenarios	LFS			WE		LFSWE			Final prediction
↓ Model	LFS1 (%)	LFS2 (%)	LFS3 (%)	TF-IDF (%)	CV (%)	CV + LFS1 (%)	CV + LFS2 (%)	CV + LFS3 (%)	Voting classifier (%)
KNN	79.6	81.8	80.6	89.6	88.2	90.3	90.5	90.1	90.16
SVM	82.5	83.5	85.6	94.5	95.61	95.6	96.1	95.01	96.73
NB	79.2	77.3	79.8	91.02	91.03	91.05	91.08	92.01	92.12
DT	81.4	79.6	80.8	89.54	89.51	90.1	89.61	89.68	89.92
Bagging	83.4	84.2	84.2	95.12	95.04	95.08	95.3	95.3	95.31
Adaboost	81.8	81.9	80.6	93.78	94.9	95.18	95.18	95.2	95.32

TABLE X
WELFAKE ACCURACY ANALYSIS ON DIFFERENT TRAINING
AND TESTING DATA SETS

Training-testing data ratios (%)	Model					
	KNN	SVM	NB	DT	Bagging	Adaboost
60 – 40	89.64	95.23	92.01	88.97	95.30	95.12
70 – 30	90.16	96.73	92.12	89.92	95.31	95.32
80 – 20	90.16	96.71	92.12	89.91	95.32	95.30
90 – 10	90.15	96.73	92.12	89.92	95.30	95.30

TABLE XI
EVALUATION RESULTS FOR DIFFERENT ML MODELS

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
KNN	90.16	89.02	90.55	89.78
SVM	96.73	94.60	98.61	96.56
NB	92.12	91.45	92.25	91.85
DT	89.92	86.10	92.62	89.24
Bagging	95.31	91.78	98.46	95.00
Adaboost	95.32	91.81	98.46	95.02

C. Comparative Text Classification

Fig. 6 compares the accuracy and F1-score of the WELFake model with the CNN and BERT state-of-the-art methods. The WELFake model achieved a 96.73% accuracy, while CNN and BERT achieved a maximum accuracy only up to 92.48% and 93.79%, respectively. Similarly, WELFake also shows a better F1-score compared to CNN and BERT due to its better generalization. While Kaliyar *et al.* [36] achieved a 98.36% accuracy using a deep NN on single data set, its accuracy reduced to 92.48% on the WELFake data set. Similarly, BERT is a pretrained model which works well with labeled data, while its performance gets compromised in a generalized data set where testing data are independent of the training data. Finally, WELFake focuses on text writing pattern linguistic features that extract the text structure and provide the syntax, sentiment, grammatical, and readability evidence specific to the text content, which explains the improved fake news detection. Overall, WELFake achieved the highest accuracy followed by BERT and CNN.

D. WELFake Generalization

We use different training and testing data sets to analyze the generalization performance of the WELFake model. For this purpose, we followed an adversarial approach that splits the WELFake data set into four constituent subsets (i.e., BuzzFeed, Reuters, McIntire, and Kaggle). We generated four experimental data sets that combine three of them as

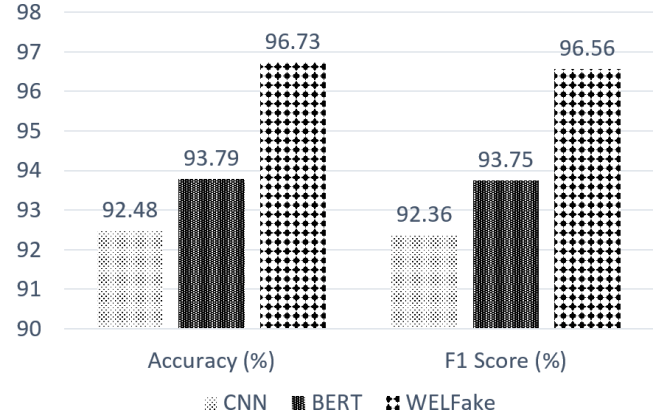


Fig. 6. CNN, BERT, and WELFake performance comparison.

TABLE XII
WELFAKE GENERALIZATION ACCURACY

No.	Training dataset		Testing dataset		Accuracy
1	BuzzFeed, Reuters, McIntire	71%	Kaggle	29%	96.70%
2	BuzzFeed, Reuters, Kaggle	91%	McIntire	9%	96.43%
3	BuzzFeed, McIntire, Kaggle	38%	Reuters	62%	96.72%
4	Reuters, McIntire, Kaggle	99%	BuzzFeed	1%	96.03%

training data and keeps the fourth for testing, as presented in Table XII.

1) *Under-Fitting Data Set (3)*: This has the training data set smaller than the testing data set, which overestimates the results in over-constrained models.

2) *Over-Fitting Data Set (4)*: This uses almost the entire data set as part of training, which underestimates the results in under-constrained models.

3) *Balanced Data Set (1, 2)*: This splits the training and testing data in acceptable ratios.

The evaluation results reveal that the WELFake model accuracy remains high in all four scenarios, ranging between 96.03% to 96.84% with a very small standard deviation of 0.3. It is possible because our proposed WELFake model handles well the overfitting and underfitting data due to the multilevel model training and data classification. These results, therefore, validate the generalization of the proposed WELFake model.

E. Related Work Comparison

We compare our proposed WELFake model with three related models, summarized in Table XIII.

TABLE XIII
WELFAKE COMPARISON WITH RELATED METHODS

Parameter	Ahmed et al. [48]	Shu et al. [49]	Gravanis et al. [37]	WELFake
Dataset	1. Kaggle	1. Politifact 2. BuzzFeed	1. Kaggle-EXT 2. BuzzFeed 3. Politifact 4. McIntire 5. UNBiased	1. Kaggle 2. BuzzFeed 3. Reuters 4. McIntire 5. WELFake
Number of news articles	1. 25,200	1. 240 2. 182	1. 23,340 2. 240 3. 182 4. 6,310 5. 3,404	1. 20,800 2. 101 3. 44,898 4. 6,335 5. 72,134
Linguistic features	No	Yes	Yes	Yes
WE	TF-IDF	No	Word2Vec	CV
Other features	No	No	NA	Voting classifier
Number of features	1	2	57	20
Classifier	Linear SVM	SVM	SVM	SVM
Accuracy	1. 92%	1. 87.8% 2. 86.4%	1. 99.0% 2. 72.7% 3. 84.7% 4. 81.0% 5. 95.0%	1. 92.60% 2. 82.78% 3. 92.04% 4. 91.78% 5. 96.73%

1) *Ahmed et al. [48]*: experimented with fake news detection on the Kaggle-EXT data set with 25 200 articles. They did not use linguistic features and applied a linear SVM model on TF-IDF with the highest 92% accuracy.

2) *Shu et al. [49]*: used linguistic features for fake news detection on the BuzzFeed and Politifact data sets of only 240 and 182 articles. They separately used a linguistic feature method with two features and implemented SVM on both data sets with an accuracy of 87.8% for Politifact and 86.4% for BuzzFeed.

3) *Gravanis et al. [37]*: used the UNBiased data set with 3404 articles with 2004 real news and 1400 fake ones and achieved accuracy up to 95% using the SVM classifier. They used 57 linguistic features with the Word2vec WE method. They also compared their method on other data sets (i.e., Kaggle-EXT, BuzzFeed, Politifact, and McIntire) and achieved an accuracy of up to 99.0%, 72.70%, 84.7%, and 81%, respectively. They claimed of using a biased Kaggle-EXT data set (in the year 2018) with real news from one source only, which produced a constant performance regardless of the number of features (i.e., up to 99% accuracy even by using a single linguistic feature, such as a typo).

4) *WELFake*: applied on a larger data set with over 72 000 news achieved a higher accuracy of 96.73% compared to the related methods. For a fair comparison, we separately applied the WELFake model on the four smaller data sets (i.e., Kaggle, McIntire, Reuters, Buzzfeed) too. Compared to [37] (the better-proven method among [37], [48], [49]) on the Buzzfeed and McIntire data sets, WELFake improved the accuracy from 72.7% to 82.7%, respectively, from 81% to 91.78%.

VIII. CONCLUSIONS AND FUTURE WORK

We presented a new model called WELFake for text fake news detection. For this purpose, we prepared a larger data set called WELFake with over 72 000 news articles combining four open-source data sets (i.e., Kaggle, McIntire, Reuters, and BuzzFeed) to reduce their individual limitation and bias. Afterward, we analyzed over 80 linguistic features from state-of-the-art works and selected 20 significant ones to minimize the

computational complexity and increase the standard classifiers' accuracy. We applied two WE-based methods (i.e., TF-IDF, CV) over these linguistic features using six ML models (i.e., KNN, SVM, NB, DT, Bagging, and AdaBoost) and found out that CV produces better overall accuracy than TF-IDF with an SVM model. We, therefore, used CV over LFS and classified the 20 features based on four categories: writing pattern, readability index, psycho-linguistics, and quantity.

As the number of predictors that participate in the voting classifier needs to be odd, we prepared three LFS by distributing the twenty selected features in a balanced manner across these categories. Afterward, we embedded CV with these LFS and applied all six ML models. We determined the most accurate ML model and took its predicted results from each WE-enabled LFS data set for voting classification. We finally applied the result of this voting classifier to the next level voting classification with the best model results of TF-IDF and CV over LFS and obtained the final classification. Experimental results show that the WELFake model produces a high 96.73% accuracy on the WELFake data set. To further analyze its advantage we compared it with two state-of-the-art works and found out that it improves the overall accuracy by 1.31% compared to BERT and 4.25% compared to CNN models. The proposed WELFake model also improved the accuracy by up to 10% on the McIntire and BuzzFeed data sets [37]. We also analyzed the performance of different ML models in terms of accuracy, precision, recall, and F1-score, and found out that SVM produced the most accurate results.

Finally, our frequency-based model focused on analyzing writing patterns outperformed predictive-based related works implemented using the Word2vec WE method by up to 1.73%.

We plan to extend our work in the future with other factors like knowledge graphs and user credibility for further verification of the output generated by the WELFake model.

REFERENCES

- [1] W. Jiang, J. Wu, F. Li, G. Wang, and H. Zheng, "Trust evaluation in online social networks using generalized network flow," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 952–963, Mar. 2016.
- [2] M. Alrubaian, M. Al-Qurishi, A. Alamri, M. Al-Rakhami, M. M. Hassan, and G. Fortino, "Credibility in online social networks: A survey," *IEEE Access*, vol. 7, pp. 2828–2855, 2019.
- [3] S. Ranganath, S. Wang, X. Hu, J. Tang, and H. Liu, "Facilitating time critical information seeking in social media," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2197–2209, Oct. 2017.
- [4] Z. Zhang, R. Sun, X. Wang, and C. Zhao, "A situational analytic method for user behavior pattern in multimedia social networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 520–528, Dec. 2019.
- [5] M. Schudson and B. Zelizer, "Fake news in context," in *Understanding and Addressing the Disinformation Ecosystem*. Philadelphia, PA, USA: Annenberg School for Communication, Apr. 2017, pp. 1–4.
- [6] S. Zaryan, "Truth and trust: How audiences are making sense of fake news," M.S. thesis, Media Commun. Studies, Lund Univ. Publications Student Papers, Stockholm, Sweden, Jun. 2017. [Online]. Available: <https://lup.lub.lu.se/student-papers/search/publication/8906886>
- [7] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, Mar. 2018.
- [8] R. Sequeira, A. Gayen, N. Ganguly, S. K. Dandapat, and J. Chandra, "A large-scale study of the Twitter follower network to characterize the spread of prescription drug abuse tweets," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 6, pp. 1232–1244, Dec. 2019.
- [9] *Politifact News Dataset*. Accessed: Mar. 31, 2020. [Online]. Available: <http://www.politifact.com/>
- [10] *The Washingtonpost Fact Checker*. Accessed: Mar. 31, 2020. [Online]. Available: <https://www.washingtonpost.com/news/fact-checker>

- [11] *Fact Check*. Accessed: Mar. 31, 2020. [Online]. Available: <https://www.factcheck.org/>
- [12] *Snopes*. Accessed: Mar. 31, 2020. [Online]. Available: <https://www.snopes.com/>
- [13] *Truthorfiction*. Accessed: Mar. 31, 2020. [Online]. Available: <https://www.truthorfiction.com/>
- [14] *Fullfact*. Accessed: Mar. 31, 2020. [Online]. Available: <https://fullfact.org/>
- [15] *Hoax Slayer*. Accessed: Mar. 31, 2020. [Online]. Available: <http://hoax-slayer.com/>
- [16] *Viswas News*. Accessed: Mar. 31, 2020. [Online]. Available: <http://www.vishvasnews.com/>
- [17] *Factly*. Accessed: Mar. 31, 2020. [Online]. Available: <https://factly.in/>
- [18] L.-L. Shi *et al.*, "Human-centric cyber social computing model for hot-event detection and propagation," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 5, pp. 1042–1050, Oct. 2019.
- [19] M. Glenski, T. Weninger, and S. Volkova, "Propagation from deceptive news sources who shares, how much, how evenly, and how quickly?" *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 4, pp. 1071–1082, Dec. 2018.
- [20] E. Lancaster, T. Chakraborty, and V. S. Subrahmanian, "MALT^P : Parallel prediction of malicious tweets," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 4, pp. 1096–1108, Dec. 2018.
- [21] P. K. Verma and P. Agrawal, "Study and detection of fake news: P²C²-based machine learning approach," in *Proc. Int. Conf. Data Manage., Anal. Innov.*, vol. 1175. Singapore: Springer, Sep. 2020, pp. 261–278.
- [22] Z. Jin, J. Cao, Y. Zhang, J. Zhou, and Q. Tian, "Novel visual and statistical image features for microblogs news verification," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 598–608, Mar. 2017.
- [23] B. Ratner, "The correlation coefficient: Its values range between +1/−1, or do they?" *J. Targeting, Meas. Anal. Marketing*, vol. 17, no. 2, pp. 139–142, Jun. 2009.
- [24] A. De Salve, P. Mori, B. Guidi, and L. Ricci, "An analysis of the internal organization of Facebook groups," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 6, pp. 1245–1256, Dec. 2019.
- [25] P. K. Verma, P. Agrawal, and R. Prodan, *WELFake Dataset for Fake News Detection in Text Data (Version: 0.1) [Data Set]*. Genève, Switzerland: Zenodo, 2021.
- [26] V. Madaan and A. Goyal, "Predicting ayurveda-based constituent balancing in human body using machine learning methods," *IEEE Access*, vol. 8, pp. 65060–65070, 2020.
- [27] M. Li, G. Clinton, Y. Miao, and F. Gao, "Short text classification via knowledge powered attention with similarity matrix based CNN," 2020, *arXiv:2002.03350*. [Online]. Available: <http://arxiv.org/abs/2002.03350>
- [28] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification," in *Proc. China Nat. Conf. Chin. Comput. Linguistics*, vol. 11856. Cham, Switzerland: Springer, Feb. 2020, pp. 194–206. [Online]. Available: <https://arxiv.org/abs/1905.05583>
- [29] K. Dzmitry Bahdanau, Y. Cho, and Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15. [Online]. Available: <https://arxiv.org/abs/1409.0473>
- [30] Y. Chen, N. J. Conroy, and V. L. Rubin, "Misleading online content: Recognizing clickbait as 'false news,'" in *Proc. ACM Workshop Multimodal Deception Detection*, Nov. 2015, pp. 15–19.
- [31] P. Bourgonje, J. Moreno Schneider, and G. Rehm, "From clickbait to fake news detection: An approach based on detecting the stance of headlines to articles," in *Proc. EMNLP Workshop, Natural Lang. Process. Meets Journalism*, 2017, pp. 84–89.
- [32] H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi, "Truth of varying shades: Analyzing language in fake news and political fact-checking," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 2931–2937.
- [33] M. Alrubaian, M. Al-Qurishi, M. Mehedi Hassan, and A. Alamri, "A credibility analysis system for assessing information on Twitter," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 4, pp. 661–674, Aug. 2018.
- [34] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on Twitter," in *Proc. 20th Int. Conf. World Wide Web (WWW)*, 2011, pp. 675–684.
- [35] D. Benjamin, D. Horne, and S. Adali, "This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news," in *Proc. 2nd Int. Workshop News Public Opinion*, Mar. 2017, pp. 1–9.
- [36] R. K. Kaliyar, A. Goswami, P. Narang, and S. Sinha, "FNDNet—A deep convolutional neural network for fake news detection," *Cognit. Syst. Res.*, vol. 61, pp. 32–44, Jun. 2020.
- [37] G. Gravanis, A. Vakali, K. Diamantaras, and P. Karadais, "Behind the cues: A benchmarking study for fake news detection," *Expert Syst. Appl.*, vol. 128, pp. 201–213, Aug. 2019.
- [38] K. Shu, L. Cui, S. Wang, D. Lee, and H. Liu, "dFEND: Explainable fake news detection," in *Proc. KDD 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 395–405.
- [39] R. Zellers *et al.*, "Defending against neural fake news," 2019, *arXiv:1905.12616*. [Online]. Available: <http://arxiv.org/abs/1905.12616>
- [40] J. K. Burgoon, J. Blair, T. Qin, and J. Nunamaker, "Detecting deception through linguistic analysis," in *Proc. 1st NSF/NIJ Conf. Intell. Secur. Inform.*, Berlin, Germany: Springer, May 2003, pp. 91–101.
- [41] M. D. Vicario, W. Quattrociocchi, A. Scala, and F. Zollo, "Polarization and fake news: Early warning of potential misinformation targets," *ACM Trans. Web*, vol. 13, no. 2, pp. 1–22, Apr. 2019.
- [42] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic detection of fake news," in *Proc. 27th Int. Conf. Comput. Linguistics*, Santa Fe, NM, USA: Association for Computational Linguistics, Aug. 2018, pp. 3391–3401.
- [43] C. Buntain and J. Golbeck, "Automatically identifying fake news in popular Twitter threads," in *Proc. IEEE Int. Conf. Smart Cloud (Smart-Cloud)*, New York, NY, USA, USA, Nov. 2017, pp. 208–215.
- [44] T. Mitra and E. Gilbert, "Credbank: A large-scale social media corpus with associated credibility annotations," in *Proc. 9th Int. AAAI Conf. Web Social Media*. Apr. 2015, pp. 258–267.
- [45] A. Zubiaga, G. W. S. Hoi, M. Liakata, and R. Procter, "PHEME dataset of rumours and non-rumours," Univ. Warwick, Coventry, U.K., Oct. 2016. [Online]. Available: https://figshare.com/articles/dataset/PHEME_dataset_of_rumours_and_non-rumours/4010619
- [46] M. L. Newman, J. W. Pennebaker, D. S. Berry, and J. M. Richards, "Lying words: Predicting deception from linguistic styles," *Personality Social Psychol. Bull.*, vol. 29, no. 5, pp. 665–675, May 2003.
- [47] L. Zhou, J. K. Burgoon, J. F. Nunamaker, and D. Twitchell, "Automating linguistics-based cues for detecting deception in text-based asynchronous computer-mediated communications," *Group Decis. Negotiation*, vol. 13, no. 1, pp. 81–106, Jan. 2004.
- [48] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using N-gram analysis and machine learning techniques," in *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, vol. 10618. Cham, Switzerland: Springer, Oct. 2017, pp. 127–138.
- [49] K. Shu, S. Wang, and H. Liu, "Exploiting Tri-relationship for fake news detection," Dec. 2018, *arXiv:1712.07709v1*. [Online]. Available: <https://arxiv.org/abs/1712.07709v1>
- [50] C. Burfoot and T. Baldwin, "Automatic satire detection: Are you having a laugh?" in *Proc. ACL-IJCNLP Conf. Short Papers ACL-IJCNLP*, 2009, pp. 161–164.
- [51] B. Riedel, I. Augenstein, G. P. Spithourakis, and S. Riedel, "A simple but tough-to-beat baseline for the fake news challenge stance detection task," 2017, *arXiv:1707.03264*. [Online]. Available: <http://arxiv.org/abs/1707.03264>
- [52] W. Y. Wang, "'Liar, liar pants on fire': A new benchmark dataset for fake news detection," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Short Papers)*, vol. 2, 2017, pp. 422–426.
- [53] *Mcintire Fake News Dataset*. Accessed: Apr. 15, 2020. [Online]. Available: <https://github.com/lutzhamel/fake-news>
- [54] *Fake News Kaggle Dataset*. Accessed: Apr. 15, 2020. [Online]. Available: <https://www.kaggle.com/c/fake-news/data?select=train.csv>
- [55] *Benjamin Political News Dataset*. Accessed: May 15, 2020. [Online]. Available: <https://github.com/rpitrust/fakenewsdata1>
- [56] *Burfoot Satire News Dataset*. Accessed: May 15, 2020. [Online]. Available: <http://www.csse.unimelb.edu.au/research/lt/resources/satire>
- [57] *Buzzfeed News Dataset*. Accessed: May 15, 2020. [Online]. Available: <https://github.com/BuzzFeedNews/2016-10-facebook-fact-check/tree/master/data>
- [58] *Credbank Dataset*. Accessed: May 15, 2020. [Online]. Available: <http://compsocial.github.io/CREDBANK-data>
- [59] *Fake News Challenge Dataset*. Accessed: May 15, 2020. [Online]. Available: <https://github.com/FakeNewsChallenge/fnc-1>
- [60] *Fakenewsnet Dataset*. Accessed: May 15, 2020. [Online]. Available: <https://github.com/KaiDMMML/FakeNewsNet>
- [61] *Liar Dataset*. Accessed: May 15, 2020. [Online]. Available: https://www.cs.ucsb.edu/~william/data/liar_dataset.zip

- [62] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
- [63] G. Shmueli, N. R. Patel, and Bruce, *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel With XLMiner*. Hoboken, NJ, USA: Wiley, 2007.
- [64] *Introduction to TF-IDF*. Accessed: Jun. 15, 2020. [Online]. Available: <http://www.tfidf.com/>



Pawan Kumar Verma (Member, IEEE) received the M.Tech. degree in computer science from the Jaypee University of Information Technology, Wazirpur, India, in 2010. He is currently pursuing the Ph.D. degree with Lovely Professional University, Phagwara, India.

He has more than eight years of teaching and research experience in opinion mining, pattern recognition, image processing, and artificial intelligence. He has authored or coauthored more than eight research papers in various international conferences and journals of repute.



Prateek Agrawal received the Ph.D. degree from IKG-Punjab Technical University, Jalandhar, India, in 2018.

He is currently a Post-Doctoral Researcher with the University of Klagenfurt, Klagenfurt, Austria, and an Associate Professor with the School of Computer Science Engineering, Lovely Professional University, Phagwara, India. He has authored or coauthored more than 60 research papers in various peer-reviewed journals and conferences.

His research interests include natural language processing, machine learning, image processing, scheduling, and parallel processing.



Ivone Amorim received the Ph.D. degree in computer science from the University of Porto, Porto, Portugal, in 2016.

She currently is a Researcher with MOG Technologies, Moreira, Portugal. She is an Associated Member of the CMUP Mathematical Research Center, University of Porto. She participated in several national and international projects. She researched in the last years at several Portuguese institutions in the area of mathematics and its applications to several fields like robotics, power systems, and cryptography.



Radu Prodan (Member, IEEE) received the Ph.D. degree from the Vienna University of Technology, Vienna, Austria, in 2004.

He was an Associate Professor with the University of Innsbruck, Innsbruck, Austria, until 2018. He is currently Professor in distributed systems with the Institute of Software Technology, University of Klagenfurt, Klagenfurt, Austria. He is the Coordinator of the European Horizon 2020 project ARTICONF that researches a toolset for trustworthy, resilient, and globally sustainable decentralized applications, with a special focus on social media networks. He was involved in numerous national and European projects with a total budget of over six million. He has authored or coauthored more than 200 publications.

Dr. Prodan was a recipient of the IEEE best paper awards.