A PROJECT REPORT

On

Comparative Analysis of Algorithms for identifying false news

submitted in partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

19WH1A0584	**Sree Aiswarya Thotakura**
19WH1A0589	**Shaik Maseera Firdose**
19WH1A05A0	**Mounika Dharmana**

Under the esteemed guidance of

Ms. B. Nagaveni

Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

June, 2023

DECLARATION

We hereby declare that the work presented in this project entitled **"Comparative Analysis of Algorithms for identifying false news"** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering For Women', Hyderabad is an authentic record of our original work carried out under the guidance of Nagaveni B, Assistant Professor, Department of CSE.

Sree Aiswarya Thotakura
(19WH1A0584)

Shaik Maseera Firdose
(19WH1A0589)

Mounika Dharmana
(19WH1A05A0)

BVRIT HYDERABAD
College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Work report on **"COMPARATIVE ANALYSIS OF ALGORITHMS FOR IDENTIFYING FALSE NEWS**" is a bonafide work carried out by Sree Aiswarya Thotakura (19WH1A0584); Shaik Maseera Firdose (19WH1A0589); Mounika Dharmana (19WH1A05A0) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department
Dr. E. Venkateswara Reddy
Professor and HoD,
Department of CSE

Guide
Ms. Nagaveni B
Assistant Professor

External Examiner

ACKNOWLEDGEMENTS

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. E. Venkateswara Reddy, Professor & HOD**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. Nagaveni B, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement, and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. SREE AISWARYA THOTAKURA
(19WH1A0584)

Ms. SHAIK MASEERA FIRDOSE
(19WH1A0589)

Ms. MOUNIKA DHARMANA
(19WH1A05A0)

LIST OF CONTENTS

ABSTRACT

False news is a problem that has grown significantly in the current digital era. Consequently, with the widespread use of social media for content dissemination, it is common to discover false and altered material that is disseminated there. A major negative effect on people and society could result from the widespread dissemination of erroneous information. It is preferable to analyze the social reactions to a piece in addition to its literary elements when determining if it is real or fake. For a particular dataset, the literacy algorithms are trained with a variety of hyperparameters to reach the highest level of delicacy. The spread of false information has led to a growing need for accurate and efficient methods to detect it. This study used machine learning algorithms to estimate how accurate fake news would be. The algorithms were trained and tested using a dataset of diverse news stories that had been classified as fake or genuine. Different machine learning models' efficacy was evaluated, including Random Forest, Support Vector Machine, and Naive Bayes and more. By detecting and filtering out fake news, the credibility of news sources can be maintained, the spread of misinformation can be prevented, harmful consequences can be avoided, healthy discourse can be promoted, and democracy can be protected.

LIST OF FIGURES

1. INTRODUCTION

In present digital age, Due to the growth of social media and online news, it is getting harder and harder to tell the difference among real news and false news. The term “Fake news refers to data that is published intentionally to confuse viewers and influence their attitudes. It takes many forms, including clickbait headlines, fabricated stories, manipulated photos and videos, and biased reporting.

Social media platforms are particularly influential, with over five hundred million tweets sent each day, offering an ideal environment for the dissemination of news with few restrictions and limitations Consequently, there is a critical need to create reliable fake news detection systems that can distinguish between real and fake news accurately.

One approach to detecting fake news involves fact-checking by humans and computer algorithms. Human fact-checkers often perform research, analyze sources, and inspect supporting documentation to confirm the truth of news pieces . Automated algorithms, on the other hand, utilize methods for natural language processing (NLP) to review the papers' content. The use of natural language in computer-human interaction is the subject of the artificial intelligence subfield known as natural language processing (NLP). The ultimate goal of NLP is to read, understand, and interpret natural languages in a useful way.

Machine learning is important for detecting fake news due to the massive volume of news stories and the speed at which they spread online. ML algorithms can quickly analyze large datasets of known fake news stories and identify patterns indicative of falsehoods, such as emotive language, sensational headlines, and lack of credible sources.

Once trained, the algorithm can classify the latest news stories as real or fake and help identify the sources of fake news and how it spreads online. ML is a powerful tool in the fight against misinformation, and it is increasingly being used by media outlets and fact-checking organizations.

Furthermore, ML can also continuously learn and adapt to new forms of fake news, as those spreading false information are constantly developing new tactics to deceive people. By using ML to detect and combat fake news, we can ensure that accurate and reliable information is disseminated and prevent the spread of harmful misinformation.

Before data is fed into models, it undergoes pre-processing techniques, including stop-word removal, lemmatization, tokenization, and vectorization. These techniques are all part of the NLP approach to natural language processing. The pre-processed data is then fed into various models, and their performance is compared and refined until the most accurate fake news detection system is achieved.

Overall, detecting fake news is a critical task in today's society, and there is a need for continuous research and development of more advanced and effective techniques to combat this problem. In order to understand and halting the spread of fake news, a combination of human fact-checking and computer algorithms using NLP techniques can be extremely effective.

1.1 Objective:

The objective of this project is to accurately identify the fake news. Our study explores different textual properties that can be used to distinguish fake content from real. By using those properties, we train a combination of different machine learning algorithms and evaluate their performance on a real-world dataset.

1.2 Methodology:

1.2.1 Dataset

The dataset is the collection of about 20800 news articles. It has been created by the University of Tennessee's Machine Learning Club, USA.
This dataset is freely available at https://www.kaggle.com/competitions/fake-news/data
Dataset consists of the following attributes

- **id**: unique id for a news article
- **title**: the title of a news article
- **author**: author of the news article
- **text**: the text of the article; could be incomplete
- **label**: a label that marks the article as potentially unreliable
    - 1: False news or unreliable
    - 0: True news or reliable

It has 10387 false news articles and 10413 true news articles. This is a good characteristic of the dataset. It shall help models give unbiased decisions.

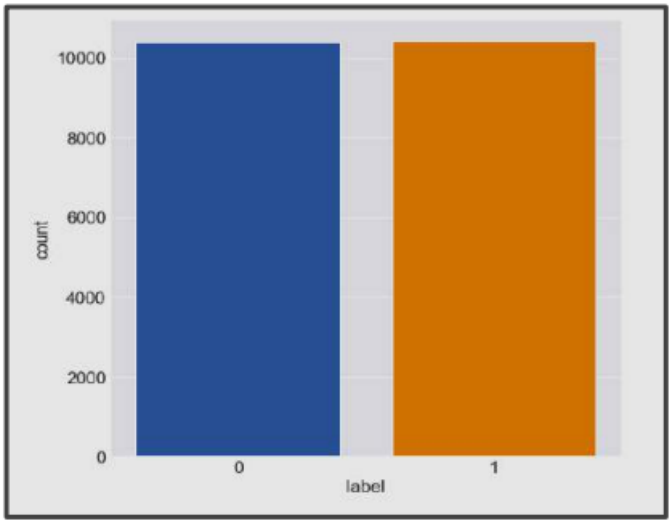

Fig 1: Dataset

1.2.2 Proposed System:

By offering a comparative review of several machine learning techniques, the suggested methodology seeks to advance discipline fake news detection. The system's results will help identify the productive algorithm in locating misleading papers and provide insights for further research in this area.

1.3 Organization of project:

Fake news detection emerged as a critical field in response to the rapid spread of misinformation and fabricated news stories, particularly in the digital era. Fake news refers to intentionally false or misleading information presented as factual news, often created and disseminated through various media platforms, including social media. It can have significant negative impacts on individuals, societies, and democratic processes.

Fake news detection involves the use of advanced technologies, such as natural language processing and machine learning algorithms, to analyze and classify news articles,

distinguishing between genuine and fake information. By employing these techniques, researchers and organizations aim to enhance the credibility of news sources, prevent the spread of misinformation, mitigate potential harm, promote healthy discourse, and safeguard the foundations of democracy.

Fake news detection using machine learning has emerged as a promising approach to combat the spread of misinformation. Machine learning algorithms can be trained on large datasets of real and fake news stories, enabling them to learn patterns and characteristics that distinguish between genuine and fabricated information. These algorithms utilize various features, such as word frequencies, linguistic patterns, and contextual cues, to classify news articles as either real or fake. Through the iterative process of training, fine-tuning, and evaluation, machine learning models can achieve high accuracy in detecting fake news.

The advantage of using machine learning lies in its ability to analyze vast amounts of data efficiently and adapt to evolving tactics employed by purveyors of fake news. However, it is essential to continuously refine and improve these models, considering the dynamic nature of misinformation and the need to address new challenges as they arise. Ultimately, machine learning plays a vital role in empowering researchers, journalists, and platforms to identify and mitigate the harmful effects of fake news on society.

The impact of fake news on society is far-reaching and multifaceted. The widespread dissemination of false or misleading information can erode trust in traditional news sources and undermine the credibility of journalism as a whole. It has the potential to manipulate public opinion, influence elections, and exacerbate social divisions. Fake news can create confusion, sow doubt, and distort public discourse on important issues such as politics, health, and science. It can also have real-world consequences, leading to misinformation-driven actions, social unrest, and even violence. Moreover, the viral nature of fake news on social media platforms amplifies its reach and impact, making it challenging to contain and debunk. Combating fake news and promoting media literacy are crucial in ensuring an informed society capable of making well-informed decisions and maintaining the integrity of public discourse.

The comparison of machine learning algorithms is crucial in the context of fake news detection to identify the most effective approach. Various machine learning models, such as

Random Forest, Support Vector Machine (SVM), Naive Bayes, and others, are evaluated to determine their efficacy in accurately classifying fake and genuine news. These algorithms differ in their underlying principles and learning strategies, which impact their performance. By comparing the accuracy, precision, recall, F1 score, or AUC-ROC of different algorithms, researchers can assess their strengths and weaknesses. For instance, SVM has shown to achieve high accuracy in fake news detection due to its ability to handle high-dimensional feature spaces and capture complex decision boundaries. Random Forest, on the other hand, excels in handling large datasets and provides robustness against overfitting. Naive Bayes, with its simplicity and efficiency, offers a competitive solution. Through this comparative analysis, researchers can identify the most suitable machine learning algorithm(s) for fake news detection, enabling the development of effective tools and systems to combat the spread of misinformation.

2. LITERATURE REVIEW

Title: A Taxonomy of Fake News Classification Techniques: Survey and Implementation Aspects

Authors: Dhiren Rohera, Harshal Shethna, Keyur Patel, Urvish Thakker, Sudeep Tanwar**,** Rajesh Gupta, Wei-Chiang Hong, Ravi Sharma

Summary: Social media has fascinated people worldwide in spreading fake news due to its easy availability, cost-effectiveness, and ease of information sharing. Thus, to mitigate the awful consequences of fake news, several research types have been conducted for its detection with high accuracy to prevent its fatal outcome. Motivated by the aforementioned concerns, we present a comprehensive survey of the existing fake news identification techniques in this paper. Then, we select Machine Learning (ML) models such as Long-Short Term Memory (LSTM), Passive Aggressive Algorithm, Random Forest (RF), and Naive Bayes (NB) and train them to detect fake news articles on the self-aggregated dataset. The model is trained on 6335 news articles, with LSTM showing the highest accuracy of 92.34% in predicting fake news and NB were showing the highest recall. Based on these results, we propose a hybrid fake news detection technique using NB and LSTM

Title: All Your Fake Detector are Belong to Us: Evaluating Adversarial Robustness of Fake-News Detectors Under Black-Box Settings

Authors: Hassan Ali, Muhammad Suleman Khan, Amer Alghadhban, Meshari Alazmi**,** Ahmad Alzamil, Khaled Al-Utaibi

Summary: One promising solution for countering this threat is to leverage deep learning (DL)-based text classification methods for fake-news detection. However, since such methods have been shown to be vulnerable to adversarial attacks, the integrity and security of DL-based fake news classifiers are under question. Although many works study text classification under the adversarial threat, to the best of our knowledge, they do not find any work in literature that specifically analyzes the performance

DL-based fake-news detectors under adversarial settings. They investigated the robustness of four different DL architectural choices—multilayer perceptron(MLP), convolutional neural network (CNN),recurrent neural network (RNN) and a recently proposed Hybrid CNN-RNN trained on three different state-of-the-art datasets— under different adversarial attacks (Text Bugger, Text Fooler, PWWS, and Deep Word Bug) implemented using the state-of-the-art NLP attack library, Text-Attack.

Title: OPCNN-FAKE: Optimized Convolutional Neural Network for Fake News Detection

Authors: Hager Saleh,Abdullah Alharbi,Saeed Hamood Alsamhi

Summary: This paper proposes novel approaches based on Machine Learning (ML) and Deep Learning (DL) for the fake news detection system to address this phenomenon. The main aim of this paper is to find the optimal model that obtains high accuracy performance. Therefore, they propose an optimized Convolutional Neural Network model to detect fake news (OPCNN-FAKE). They compare the performance of the OPCNN-FAKE with Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and The six regular ML techniques: Decision Tree (DT), logistic Regression (LR), K Nearest Neighbor (KNN), Random Forest (RF), Support Vector Machine (SVM), and Naive Bayes (NB) using four fake news benchmark datasets. To evaluate the performance of the OPCNN-FAKE, accuracy, precision, recall, F1-measure were applied to validate the results. The results show that OPCNN-FAKE model has achieved the best performance for each dataset compared with other models

Title: Detecting fake news over online social media via domain reputations and content understanding

Authors: Kuai Xu,Feng Wang,Haiyan Wang,Bo Yang

Summary: As a first step of fighting with fake news, this paper characterizes hundreds of popular fake and real news measured by shares, reactions, and comments on Facebook from two perspectives:

domain reputations and content understandinOur domain reputation analysis reveals that the Web sites of the fake and real news publishers exhibit diverse registration behaviors, registration timing, domain rankings, and domain popularity. To the best of our knowledge, this is the first effort to systematically study domain reputations and content characteristics of fake and real news, which will provide key insights for effectively detecting fake news on social media.

Title: A Novel Stacking Approach for Accurate Detection of Fake News

Authors: Tao Jiang,Jian Ping Li,Amin Ul Haq,Abdus Saboor,Amjad Ali

Summary: News online has become the major source of information for people. Some fake news are so similar to the real ones that it is difficult for human to identify them. Therefore, automated fake news detection tools like machine learning and deep learning models have become an essential requirement. In this paper, they evaluated the performance of five machine learning models and three deep learning models on two fake and real news datasets of different size with hold out cross validation. They also used term frequency, term frequency-inverse document frequency and embedding techniques to obtain text representation for machine learning and deep learning models respectively.

Title: Fake News Stance Detection Using Deep Learning Architecture (CNN-LSTM)

Authors: Muhammad Umer, Zainab Imtiaz, Saleem Ullah, Arif Mehmood, Gyu Sang Choi, Byung-Won On

Summary: To address the aforementioned issue, a hybrid Neural Network architecture, that combines the capabilities of CNN and LSTM, is used with two different dimensionality reduction approaches, Principle Component Analysis (PCA) and Chi-Square. This work proposed to employ the dimensionality reduction techniques to reduce the dimensionality of the feature vectors before passing them to the classifier. The nonlinear features are fed to PCA and chi-square which provides more contextual features for fake news detection.

The experimental results show that PCA outperforms than Chi-square and state-of-the-art methods. In this paper, they propose a model that automatically classifies the news articles with stance labels of either agree, disagree, unrelated, or discuss. The proposed methodology is based on the observations to find the relevance of articles, which can be found using keywords within headlines.

Title: A Comprehensive Review on Fake News Detection With Deep Learning

Authors: M. F. Mridha,Ashfia Jannat Keya,Md. Abdul Hamid,Muhammad Mostafa Monowar,Md. Saifur Rahman

Summary: A protuberant issue of the present time is that, organizations from different domains are struggling to obtain effective solutions for detecting online-based fake news. Compared with many machine learning techniques, deep learning-based techniques are capable of detecting fake news more accurately. Previous review papers were based on data mining and machine learning techniques, scarcely exploring the deep learning techniques for fake news detection. This study attempts to investigate advanced and state-of-the-art fake news detection mechanisms pensively. The prominent evaluation metrics in fake news detection are also discussed. Nevertheless, they suggest further recommendations to improve fake news detection mechanisms in future research directions.

Title: Evidence-Aware Multilingual Fake News Detection

Authors: Hicham Hammouchi, Mounir Ghogho

Summary: As a result, governments, media agencies, and academics have established factchecking units and developed automatic detection systems. Research approaches to verify the veracity of news focused largely on writing styles, propagation patterns, and building knowledge bases that serve as a reference for fact checking. However, little work has been done to assess the credibility of the source of the claim to be checked.

This paper proposes a general framework for detecting fake news that uses external evidence to verify the veracity of online news in a multilingual setting. Search results from Google are used as evidence, and the claim is cross-checked with the top five search results. All of these components are combined to derive better predictions of the veracity of claims. With the achieved results, the proposed framework present a promising automatic fact checker for both early and late detection.

Title: Unsupervised Fake News Detection Based on Autoencoder

Authors: Dun Li,Haimei Guo,Zhenfei Wang,Zhiyun Zheng

Summary: Fake news can be regarded as an anomaly on social networks, and autoencoder can be used as the basic unsupervised learning method. So, an unsupervised fake news detection method based on autoencoder (UFNDA) is proposed. Next, to obtain the hidden information and internal relationship between features, Bidirectional GRU(Bi-GRU) layer and Self-Attention layer are added into the autoencoder, and then reconstruct residual to detect fake news. This paper treats fake news as an abnormal data in social networks and proposes a method based on autoencoder for unsupervised fake news detection with various features, namely UFNDA. After training UFNDA with real news data, the UFNDA has the ability to detect fake news. By testing the test set, to obtain the reconstructed residual to distinguish real news and fake news.

3. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

3.1 Requirements Gathering:

3.1.1 Software requirements:

- Language Used: Python 3.8
- Operating System: Windows
- Software: Jupyter

3.1.2 Hardware requirements:

- Processor: Intel Core i3
- RAM: 4GB

3.2 Technologies Description:

- **Jupyter Notebook:**

Jupyter Notebook is widely used in data science, machine learning, scientific computing, and education. Its interactive and exploratory nature makes it a powerful tool for prototyping, presenting research findings, and creating reproducible data analyses. It provides a flexible and user-friendly environment for working with code, visualizations, and explanatory text in an integrated manner.

- **Libraries:**

1. NumPy:

NumPy is a powerful Python library widely used for scientific computing and data analysis. It provides a multidimensional array object that allows efficient storage and manipulation of large numerical datasets. NumPy's key feature is its ability to perform fast mathematical and logical operations on entire arrays, making it an essential tool for tasks such as linear algebra, statistical analysis, and signal processing. It also offers a range of functions for array manipulation, array indexing, and mathematical operations, enabling users to perform complex computations with ease. NumPy's intuitive and efficient syntax, along with its extensive collection of mathematical functions, makes

it a fundamental component of the scientific Python ecosystem and an invaluable resource for data scientists, researchers, and developers.

2.Pandas:

Pandas is a widely used Python library that simplifies data manipulation and analysis. It provides easy-to-use data structures, such as Data Frames, which are two-dimensional tabular structures capable of holding diverse data types. With Pandas, users can efficiently load, clean, transform, and analyze structured data from various sources. Its rich functionalities include filtering, sorting, grouping, merging, and reshaping data, along with powerful indexing and slicing capabilities. Pandas integrates seamlessly with NumPy for optimized numerical operations and with other libraries like Matplotlib and scikit-learn for data visualization and machine learning tasks. Overall, Pandas is a versatile tool that streamlines data handling, making it indispensable for data scientists, analysts, and researchers.

3.Matplotlib:

Matplotlib is a widely used Python library for creating high-quality visualizations and plots. It provides a comprehensive set of functions and tools for generating a wide range of plots, charts, histograms, and other visual representations of data. With Matplotlib, users can create static, animated, or interactive visualizations to effectively communicate insights and patterns in data. The library offers a flexible and customizable interface, allowing users to control various aspects of their plots, including axes, labels, colors, and styles. Matplotlib supports multiple plotting styles and can output visualizations in various formats, such as PNG, PDF, SVG, and more. It integrates well with other libraries in the scientific Python ecosystem, making it a valuable tool for data analysis, scientific research, and data visualization tasks. Overall, Matplotlib empowers users to create visually appealing and informative plots, enabling effective data exploration and presentation.

4.Seaborn:

Seaborn is a Python data visualization library that builds on top of Matplotlib, offering a high-level interface for creating visually appealing statistical graphics. With Seaborn,

users can easily generate a wide range of plots and charts, leveraging pre-defined themes and color palettes to enhance visual aesthetics. The library seamlessly integrates with Pandas, allowing for effortless visualization of data stored in DataFrames. Seaborn provides an extensive collection of statistical plots and offers customization options for fine-tuning visual details. It is widely utilized by data scientists, researchers, and analysts to create visually captivating and informative plots for data exploration and analysis.

5. NLTK:

NLTK (Natural Language Toolkit) is a comprehensive Python library widely used for natural language processing (NLP) tasks. It provides a collection of tools, resources, and algorithms for tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, parsing, semantic reasoning, and more. NLTK offers a wide range of corpora, lexical resources, and language models, making it a valuable resource for NLP research and application development. It also includes a set of pre-trained models and classifiers for tasks like sentiment analysis, named entity recognition, and text classification. NLTK's user-friendly interface and extensive documentation make it accessible to both beginners and experienced NLP practitioners, making it a popular choice for NLP projects and research in academia and industry.

6. Sci-kit Learn:

Scikit-learn, also known as sklearn, is a popular Python library for machine learning tasks. It offers a comprehensive set of tools and algorithms for classification, regression, clustering, and dimensionality reduction. Built on top of NumPy and SciPy, scikit-learn provides a user-friendly API, making it accessible to both beginners and experienced practitioners. With its emphasis on code readability, model interpretability, and reproducibility, scikit-learn simplifies the development of machine learning models. It integrates well with other Python libraries, enabling seamless data manipulation, visualization, and analysis. Overall, scikit-learn is a powerful and versatile library that serves as a go-to resource for various machine learning applications.

4. DESIGN

4.1 Introduction:

By offering a comparative review of several machine learning techniques, the suggested methodology seeks to advance discipline fake news detection. The system's results will help identify the productive algorithm in locating misleading papers and provide insights for further research in this area.

The system consists of the following components:

- **Data Collection:** The system will collect a large labelled dataset of news articles from various Kaggle, including both credible and false news sources.

- **Pre-processing:** The collected data will undergo pre-processing steps such as cleaning, normalization, and tokenization. Data Cleaning involved missing data imputation where every null value is replaced with white space to avoid error while training. For increasing data efficiency used NLP preprocessing techniques that includes,
    1. **Removing Special Characters using Regex:** Regex is used to remove special characters by applying pattern matching to identify and eliminate non-alphanumeric or non-whitespace characters from a string. The regex pattern can be constructed to target specific characters or character ranges, such as symbols or punctuation marks. By using regex's substitution functionality, the special characters can be replaced with an empty string, effectively removing them from the original text.
    2. **Tokenization:** Tokenization is the procedure of splitting down a stream of data into smaller units called tokens. These tokens can be simple words, phrases, or sentences, depending on the level of granularity required. It helps in converting unstructured text data into a structured format that can be easily processed and analyzed by algorithms.
    3. **Stop Words Removal:** Stopword removal is a preprocessing technique in natural language processing that involves eliminating commonly used words that carry little or no meaning in a given context. These words, known as stopwords, include articles, conjunctions, prepositions, and other frequently occurring words. Stopword removal can be performed using predefined lists of stopwords or by

creating custom lists based on the specific domain or language of the text being processed. This process helps in reducing noise and improving the quality of textual data for further analysis.

4. **Lemmatization:** Lemmatization is a linguistic procedure that involves reducing words to their base or root form, known as the lemma. lemmatization takes into account the context and meaning of words and converts them to their base form, that helps in achieving better normalization. Lemmatization considers factors such as part-of-speech and morphological analysis to ensure that words are transformed to their canonical form, making them easier to analyze and interpret. The resulting lemmas retain the semantic meaning of the original words, allowing for more meaningful and accurate analysis of textual data.

- **Feature Extraction:** The pre-processed data will be used by the system to extract pertinent features including the frequency of particular terms, the article's tone, the news source, and other metadata. In our system we used count vectorizer and Tf-idf vectorizer.
    1. **CountVectorizer:** A text preparation method that turns text documents into a token count matrix that represents the frequency of each word in the data. It is frequently employed for feature extraction in jobs involving natural language processing.
    2. **TF-IDF vectorizer:** A feature extraction method which turns text files into numerical vectors. It gives words weights based on their frequency in a document and their inverse frequency across all documents, making words in a document collection more meaningfully represented.

       Mathematical formula for Tf-Idf is given by:

       $$w_{i,j} = tf_{i,j} * \log\left(\frac{N}{df_i}\right) \quad (1)$$

       In the equation (1), tf_i is number of occurrences of i in j, df_i is number of documents containing i and N denotes total number of documents.
- **Algorithm Selection:** The system will identify the news stories as genuine or false using a variety of machine learning methods, including Naive Bayes, Support Vector Machines, Random Forest, etc.
- **Model Training:** The system will train the selected algorithms on the pre-processed data and evaluate their performance using the validation set. The algorithm with the best performance will be selected for further testing.

- **Model Evaluation:** On the testing set, the chosen method will be assessed for accuracy, precision, recall, and F1 score.
- **Comparative Analysis:** Based on their evaluation measures, the system will compare how well various machine learning algorithms perform in identifying fake news. The analysis will help identify the strengths and weaknesses of each algorithm and provide insights for further improvements.

Result Visualization: The system will visualize the performance of each algorithm using various graphs and charts to facilitate data interpretation.

4.2 Architecture Diagram:

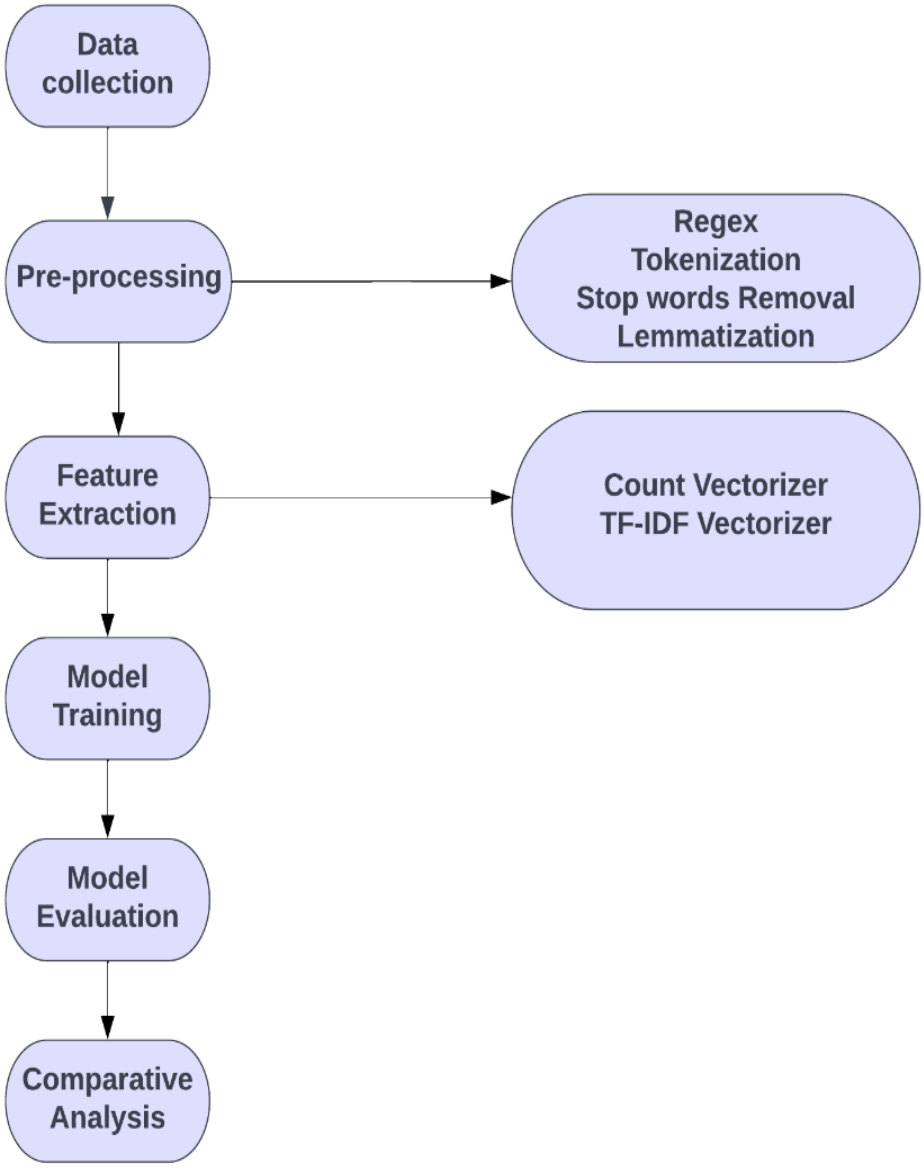

Fig 2: Workflow Design

4.3 Algorithms:

4.3.1 Support Vector Machines (SVMs):

SVMs are effective at handling high-dimensional data, making them suitable for analyzing textual data. SVMs operate by identifying the hyperplane that best divides data into various classes (see Fig 2). Real and fake news would be the classes in the fake news detection scenario. During training, the SVM algorithm learns the optimal hyperplane that separates the real and fake news samples. Once trained, the SVM model may be used to categorize new samples as either true or fake news.

SVMs have several advantages when it comes to fake news detection. They can handle large amounts of data, which is important for analyzing the vast amount of news articles and social media posts that are produced each day. Additionally, SVMs can learn complex decision boundaries, making them effective at distinguishing between real and fake news.

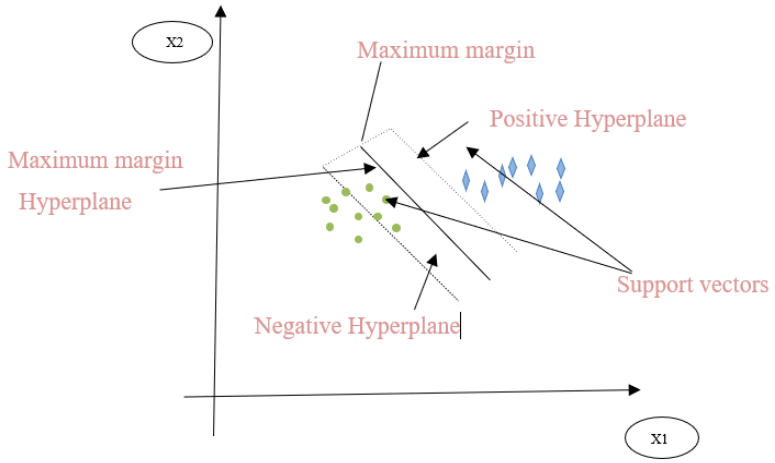

Fig 3: SVM Model

4.3.2 Decision Trees:

Popular machine learning algorithms include decision trees, which is widely used for classification problems. Decision trees are applied in the context of spotting fake news to identify the key qualities or traits that support the genuineness of news stories. The decision tree algorithm works by recursively splitting the dataset based on the most informative feature, creating a tree-like structure of decisions that leads to a final prediction (see Fig 3 The ultimate choice is made using the leaf node of the tree, which indicates whether the news article is real or fake. The advantage of decision tree algorithm for fake news detection is its interpretability, as the tree structure provides insights into the key features that distinguish between real and fake news.

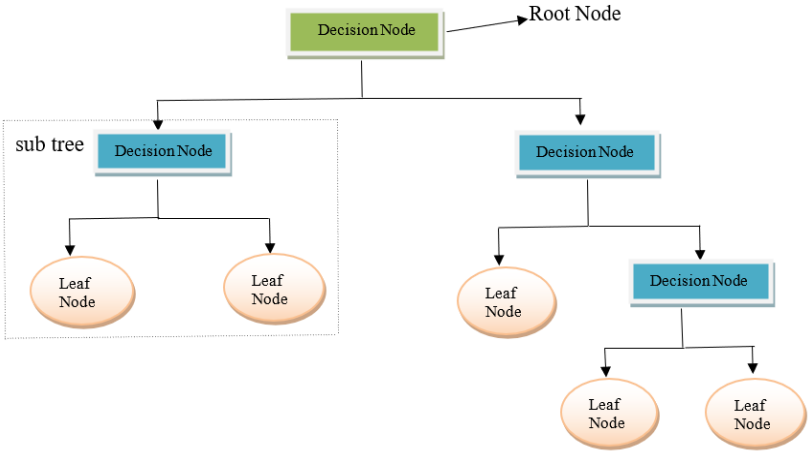

Fig 4: Decision Tree Model

4.3.3 Random Forest:

With the help of the machine learning technique known as Random Forest, the model's accuracy and robustness are increased. Multiple decision trees are built utilizing various subsets of the training data and features by the Random Forest algorithm (see Fig 4). Each decision tree, which is trained on a distinct random sample of the data and features, contributes to the final classification. This method enhances the model's capacity for generalization while reducing overfitting. Random Forest can provide insights into the importance of unique features in the classification process, which can be useful for understanding the underlying factors that contribute to fake news.

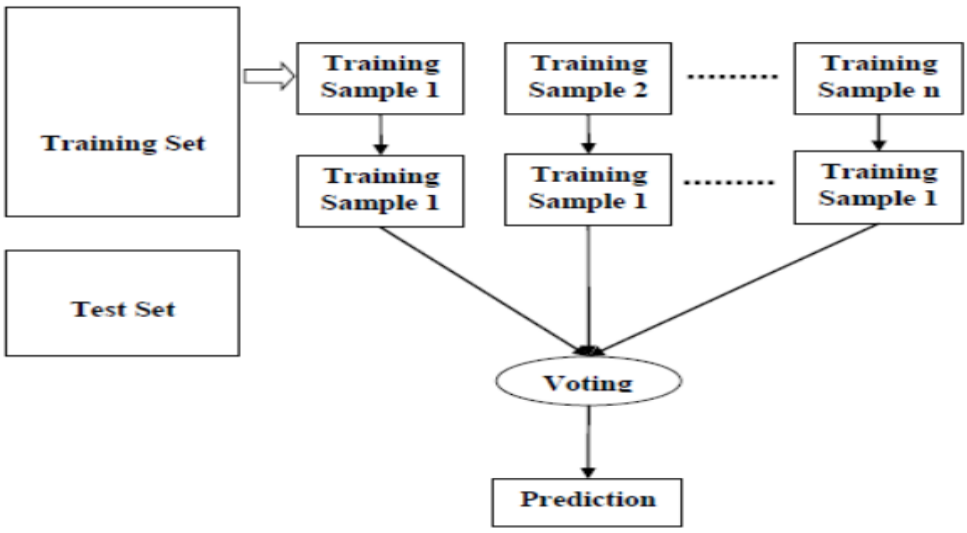

Fig 5: Random Forest Model

4.3.4 Logistic Regression:

In binary classification situations, where the objective is to predict the chances for a binary outcome, logistic regression is a statistical model that is frequently utilized. The likelihood of an article being false or authentic can be predicted using logistic regression in the context of fake news identification, a collection of attributes taken from the article. These characteristics could include, among other things, the number of emotive words, the author of the piece, or the quantity of typos.

The input features are converted by the logistic regression model into a probability value of 0 or 1, with values closer to one showing a higher likelihood that article is false. Through training on a labelled dataset of false and true news stories, the model discovers the relationship between the input attributes and the binary output.

Equation of logistic regression:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n \qquad (2)$$

4.3.5 K-Nearest Neighbors:

KNN is a popular algorithm used in machine learning for classification problems. In the context of fake news detection, KNN works by measuring the similarity between a news article and other articles in a training set. The news article is then classified using the training set's K-nearest neighbors' dominant class labels. The value of K is typically determined using cross-validation techniques.

The advantages of using KNN for fake news detection is its simplicity and ease of implementation. It does not require any assumptions about the distribution of the data and can handle nonlinear decision boundaries.

4.3.6 Multinomial NB:

Natural language processing (NLP) and text classification tasks, such as fake news identification, use the classification technique known as Multinomial Naive Bayes (MNB). It is a subtype of Naive Bayes algorithm, it performs well with features like word frequencies

in text data that have discrete and multinomial distributions. MNB makes the assumption that a word's frequency in a document is unrelated to the frequency of other terms in that document., which can be a reasonable assumption for text data.

In fake news detection, based on their word frequencies, MNB can be used to classify stories as either true or false. Here algorithm works by first training a model on a labelled dataset of news articles, where each article is assigned a label of either fake or real. The model then calculates the probabilities of each word appearing in fake and real news articles, which are used to classify new, unseen articles based on their word frequencies. The article is classified as fake or real based on which label has a higher probability given its word frequencies.

5. IMPLEMENTATION

5.1 Coding:

Git link: https://github.com/19WH1A0584/Comparative-Analysis-of-Algorithms-for-identifying-false-news.git

5.2 Training Dataset Screenshots:

The dataset Fake-news is taken from Kaggle which consists of 5 attributes and 2 classes (Fake-1, Real-0) with sizes Test- 5,200 and Train- 20,800.

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Fig 6: Sample Dataset

Evaluation metrics:

The effectiveness of our approaches were measured using several evaluation metrics, which are: Accuracy (ACC), Recall (REC), Precision (PRE) and F1-score (F1).

- **Accuracy:**(ACC)

  It is calculated by dividing the number of accurate predictions by the total number of things under consideration.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

- **Recall:**(REC)

  It relates to the proportion of positively classified items to those that are truly positive.

$$REC = \frac{TP}{TP+FN} \quad (4)$$

- **Precision:**(PRE)

  It describes how much information a number can hold in terms of its digits and illustrates how closely two or more measures are spaced from one another.

  $$PRE = \frac{TP}{TP+FP} \quad (5)$$

- **F1- Score:** (F1)

  It provides a quality metric that balances the relative value of these two measurements, it links the precision (PRE) and recall (REC) metrics.

  $$F1 = \frac{2*(PRE*REC)}{PRE+REC} \quad (6)$$

  Where TP is true positives, TN is true negatives, FP is false positives, FN is false negatives.

- **Confusion matrix:**

  A table summarizing a ML algorithm's performance is called a confusion matrix. It assesses true positives, true negatives, false positives, and false negatives by contrasting projected labels with the actual labels of a dataset. This split makes it possible to assess the model's precision, recall, accuracy, and F1 score. The confusion matrix assists in improving the performance of ML algorithms by highlighting their advantages and disadvantages.

5.3 Input Screenshots:

The dataset is taken as input in form of text as seen.

	id	title	author	text	total
0	20800	Specter of Trump Loosens Tongues, if Not Purse...	David Streitfeld	PALO ALTO, Calif. — After years of scorning...	Specter of Trump Loosens Tongues, if Not Purse...
1	20801	Russian warships ready to strike terrorists ne...		Russian warships ready to strike terrorists ne...	Russian warships ready to strike terrorists ne...
2	20802	#NoDAPL: Native American Leaders Vow to Stay A...	Common Dreams	Videos #NoDAPL: Native American Leaders Vow to...	#NoDAPL: Native American Leaders Vow to Stay A...
3	20803	Tim Tebow Will Attempt Another Comeback, This ...	Daniel Victor	If at first you don't succeed, try a different...	Tim Tebow Will Attempt Another Comeback, This ...
4	20804	Keiser Report: Meme Wars (E995)	Truth Broadcast Network	42 mins ago 1 Views 0 Comments 0 Likes 'For th...	Keiser Report: Meme Wars (E995) Truth Broadcas...

Fig 7: Input Dataset

5.4 Output Screenshots:

Confusion matrix for various ML algorithms for our system are as follows,

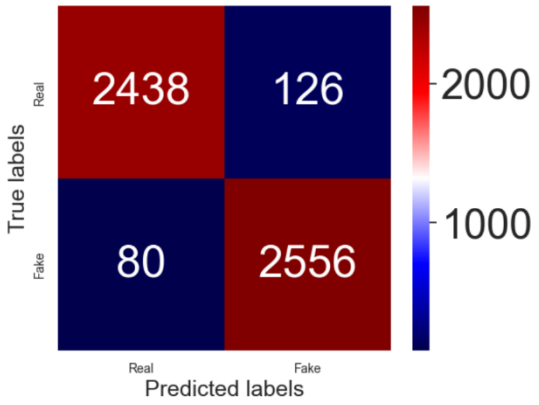

Fig 8: Confusion matrix of Decision Tree

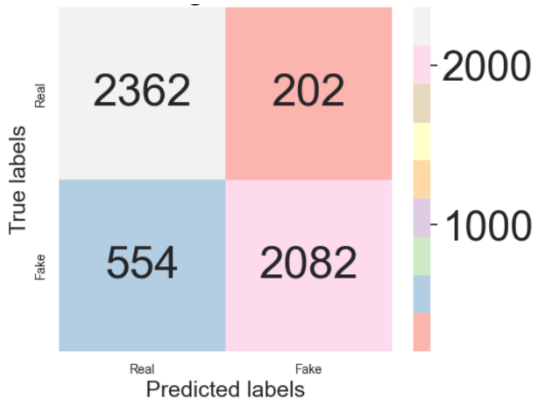

Fig 9: Confusion matrix of KNN

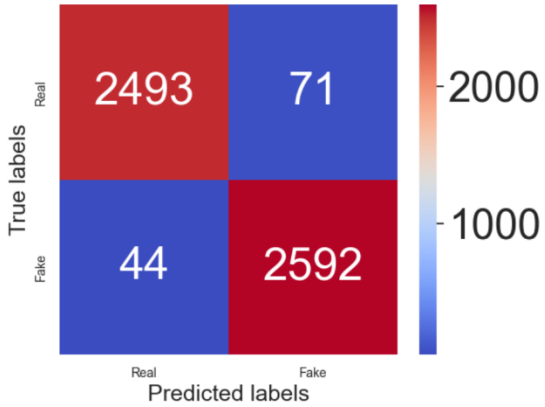

Fig 10: Confusion matrix of Logistic Regression

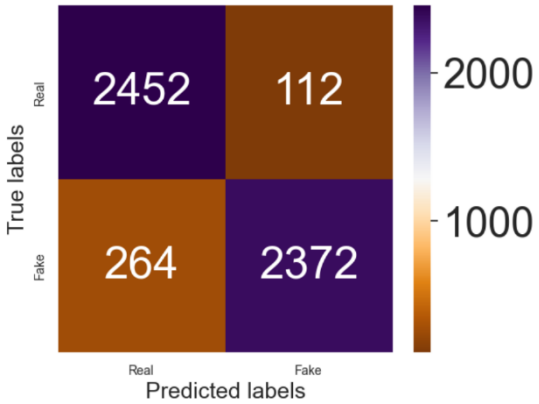

Fig 11: Confusion matrix of Random forest

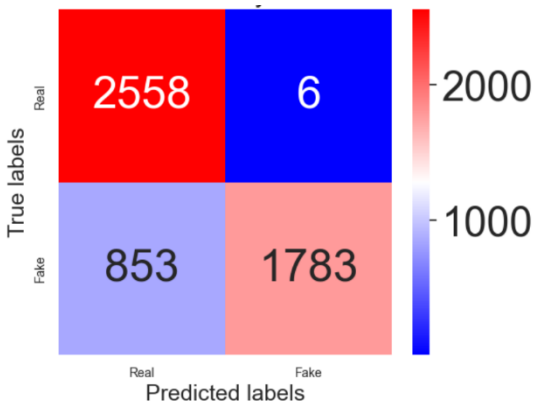

Fig 12: Confusion matrix of Multinomial NB

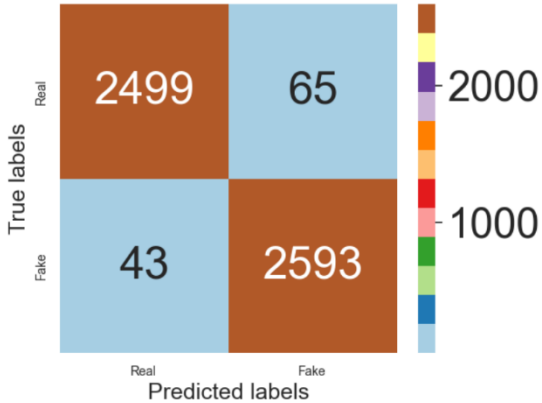

Fig 13: Confusion matrix of SVM

Based on the accuracy achieved by each algorithm we try to find out the best approach to find fake news.

Table 1. Table of Comparison.

S.NO	ALGORITHM	ACCURACY	PRECISION	RECALL	F1-SCORE
1	SVM	98	98	98	98
2	Decision Tree	96	96	96	96
3	Random Forest	93	93	93	93
4	Logistic Regression	98	98	98	98
5	KNN	86	87	86	86
6	Multinomial NB	83	87	84	83

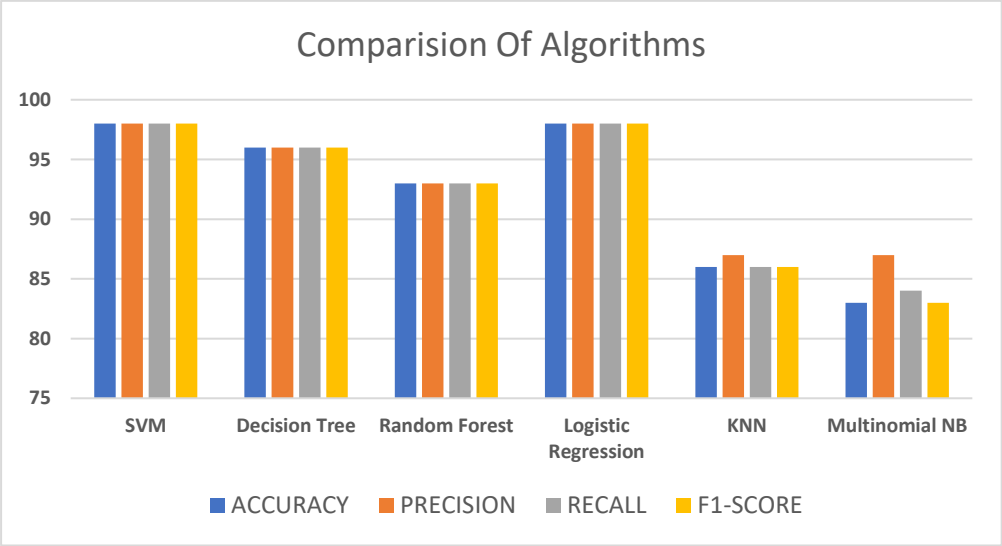

Fig 14: Comparison of Algorithms

We see that SVM and Logistic Regression models perform better in relation to recall, precision, accuracy, f1-score, and false negative values. They have rather high accuracy values and somewhat lower false negative rates. Therefore, these models are superior options for the purpose of classifying fake news.

6. CONCLUSION AND FUTURE SCOPE

These are abstracts of seven separate research works that try to create computer programs that can recognize bogus news. The research seeks to use deep learning and/or machine learning to identify and categorize erroneous material. They all acknowledge the difficulties faced by the fake news' dominance on social media. The research uses different datasets along with various techniques, such as reinforcement learning and supervised, unsupervised, and semi-supervised techniques. The studies assess the effectiveness of their models in relation to several standards, including accuracy, robustness, and precision, and they all present encouraging findings. The authors of the studies suggest that future research in this area should investigate other textual and visual features of false news, incorporate tree-based learning and multimodal approaches, and validate proposed models on larger datasets. Future work on detecting false news may combine NLP techniques with advanced machine learning strategies like reinforcement learning and deep learning for increased accuracy. Social network analysis could monitor expansion of bogus rumors on internet community. Collaboration between researchers, journalists, and social media companies could aid the development of more effective detection methods.

7. REFERENCES

[1] D. Rohera et al., "A Taxonomy of Fake News Classification Techniques: Survey and Implementation Aspects," in IEEE Access, vol. 10, pp. 30367-30394, 2022, doi: 10.1109/ACCESS.2022.3159651.

[2] H. Ali et al., "All Your Fake Detector are Belong to Us: Evaluating Adversarial Robustness of Fake-News Detectors Under Black-Box Settings," in IEEE Access, vol. 9, pp. 81678-81692, 2021, doi: 10.1109/ACCESS.2021.3085875.

[3] H. Saleh, A. Alharbi and S. H. Alsamhi, "OPCNN-FAKE: Optimized Convolutional Neural Network for Fake News Detection," in IEEE Access, vol. 9, pp. 129471-129489, 2021, doi: 10.1109/ACCESS.2021.3112806.

[4] K. Xu, F. Wang, H. Wang, and B. Yang, "Detecting fake news over online social media via domain reputations and content understanding," in Tsinghua Science and Technology, vol. 25, no. 1, pp. 20-27, Feb. 2020, doi: 10.26599/TST.2018.9010139.

[5] T. Jiang, J. P. Li, A. U. Haq, A. Saboor, and A. Ali, "A Novel Stacking Approach for Accurate Detection of Fake News," in IEEE Access, vol. 9, pp. 22626-22639, 2021, doi: 10.1109/ACCESS.2021.3056079.

[6] M. Umer, Z. Imtiaz, S. Ullah, A. Mehmood, G. S. Choi, and B. -W. On, "Fake News Stance Detection Using Deep Learning Architecture (CNN-LSTM)," in IEEE Access, vol. 8, pp. 156695-156706, 2020, doi: 10.1109/ACCESS.2020.3019735.

[7] M. F. Mridha, A. J. Keya, M. A. Hamid, M. M. Monowar, and M. S. Rahman, "A Comprehensive Review on Fake News Detection with Deep Learning," in IEEE Access, vol. 9, pp. 156151-156170, 2021, doi: 10.1109/ACCESS.2021.3129329.

[8] H. Hammouchi and M. Ghogho, "Evidence-Aware Multilingual Fake News Detection," in IEEE Access, vol. 10, pp. 116808-116818, 2022, doi: 10.1109/ACCESS.2022.3220690.

[9] D. Li, H. Guo, Z. Wang, and Z. Zheng, "Unsupervised Fake News Detection Based on Autoencoder," in IEEE Access, vol. 9, pp. 29356-29365, 2021, doi: 10.1109/ACCESS.2021.3058809.

8. APPENDIX

Sample code:

1. **Check number of NULL values in the dataset**

```
# how many null values in the dataset
print("Null values in train data:")
print(train.isnull().sum())
print('\n\n')

print("Null values in test data:")
print(test.isnull().sum())

sns.heatmap(train.isnull(),cmap='YlGnBu_r')
print(train.dtypes)
```

2. **Missing data imputation**

```
#imputing the data
test=test.fillna(' ')
train=train.fillna(' ')

from sklearn.feature_extraction.text import CountVectorizer
def get_top_n_words(corpus, n=None):
   vec = CountVectorizer().fit(corpus)
   bag_of_words = vec.transform(corpus)
   sum_words = bag_of_words.sum(axis=0)
   words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
   words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
   return words_freq[:n]
common_words = get_top_n_words(train['text'], 20)
for word, freq in common_words:
   print(word, freq)
df1 = pd.DataFrame(common_words, columns = ['text' , 'count'])
df1.groupby('text').sum()['count'].sort_values(ascending=False).plot(
kind='bar', title='Top 20 words in dataset before removing stop
words',color=['slateblue', 'blueviolet', 'violet', 'orchid', 'lightpink'])

def get_top_n_bigram(corpus, n=None):
   vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
   bag_of_words = vec.transform(corpus)
   sum_words = bag_of_words.sum(axis=0)
   words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
   words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
   return words_freq[:n]
```

```
common_words = get_top_n_bigram(train['text'], 20)
for word, freq in common_words:
    print(word, freq)
df3 = pd.DataFrame(common_words, columns = ['text' , 'count'])
df3.groupby('text').sum()['count'].sort_values(ascending=False).plot(
kind='bar', title='Top 20 bigrams in dataset before removing stop
words',color=['slateblue', 'blueviolet', 'violet', 'orchid', 'lightpink'])

train['label'].plot(
    kind='hist',
    bins=50,
    title='Number of True vs Fake News')
```

3. Merging the columns (title, author, text) into one column

```
test['total']=test['title']+' '+test['author']+test['text']
train['total']=train['title']+' '+train['author']+train['text']
train.head()
test.head()
```

4. Cleaning and pre-processing

1 Regex

```
#Remove punctuations from the String
sample = "!> NLP is $$ >^sh!!!o%%rt &&%$fo@@@r^^^&&!& >*Natural@#
Language&&\ Pro@@@##%^^&cessing!@# %%$"
# what is gonna get selected we r gonna replace that with the empty string(2nd
parameter)
sample = re.sub(r'[^\w\s]','',sample)
print(sample)
```

2 Tokenization

```
import nltk
nltk.download('punkt')
print("The NLTK tokeniser has tokenised \"Computers are not as great at
understanding words as they are numbers.\" into a list of tokens ", end="\n\n")
print(nltk.word_tokenize("Computers are not as great at understanding words as they
are numbers."))
```

3 StopWords

```
sample_text = "Does this thing really work? Lets see."
print("Priniting all the different sentences in sample_text: ", end="\n\n")
for i in nltk.sent_tokenize(sample_text):
    print(i)
words = nltk.word_tokenize(sample_text)
print("Priniting all the different words in sample_text: ", end="\n\n")
for i in nltk.word_tokenize(sample_text):
    print(i)
```

```
stop=stopwords.words("english")
print(stop)
clean_words = [w for w in words if not w in stop]
for i in clean_words:
   print(i)
words = nltk.word_tokenize(sample_text.lower())
clean_words = [w for w in words if not w in stop]
for i in clean_words:
   print(i)
punctuations = list(string.punctuation)
print(punctuations)
stop = stop + punctuations
print(stop)
clean_words = [w for w in words if not w in stop]
clean_words
```

4 Lemmatization

```
#Using WordNet lemmatizer
lemmatizer=WordNetLemmatizer()
input_str="Kites Babies Meeting Is Done Languages Cities Mice"
#Tokenize the sentence
input_str=nltk.word_tokenize(input_str)
lemmatizer = WordNetLemmatizer() #need to tokenise the complee sentence
print("Below, see how kites->kite, babies->baby, languages -> language, cities ->
city, mice -> mouse. Stemming couldn't have done this", end="\n\n")
#now each token i ll pass to he lemmatizer to see its reduced form
#Lemmatize each word
for word in input_str:
   print(lemmatizer.lemmatize(word).lower())
```

5. Applying Text Preprocessing techniques discussed above on Train data

```
lemmatizer=WordNetLemmatizer()
for index,row in train.iterrows(): #taking he train data and iterating each row
   filter_sentence = ''
   sentence = row['total']
   sentence = re.sub(r'[^\w\s]','',sentence)
   words = nltk.word_tokenize(sentence) #tokenizing the sentence
   words = [w for w in words if not w in stop]  #removing the stopwords
   #after removing the stopwords, applying the WornNet Lemmatizer
   for word in words:
      filter_sentence = filter_sentence + ' ' + str(lemmatizer.lemmatize(word)).lower()
   # at the end, again putting the filter_sentence back into the training document at the
same position
   train.loc[index,'total'] = filter_sentence
```

```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words = 'english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
common_words = get_top_n_words(train['total'], 20)
for word, freq in common_words:
    print(word, freq)
df2 = pd.DataFrame(common_words, columns = ['total' , 'count'])
df2.groupby('total').sum()['count'].sort_values(ascending=False).plot(
kind='bar', title='Top 20 words in dataset after text-preprocessing',color=['salmon',
'tomato', 'darksalmon', 'coral', 'orangered'])

def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
common_words = get_top_n_bigram(train['total'], 20)
for word, freq in common_words:
    print(word, freq)
df4 = pd.DataFrame(common_words, columns = ['total' , 'count'])
df4.groupby('total').sum()['count'].sort_values(ascending=False).plot(
kind='bar', title='Top 20 bi-grams in dataset after text-preprocessing',
color=['darkmagenta', 'orchid', 'mediumvioletred', 'deeppink', 'hotpink',
'palevioletred'])
 lis_text = [
     train[train['label'] == 0]['total'],
    train[train['label'] == 1]['total']
]
 lis_title = [
    train[train['label'] == 0]['total'],
    train[train['label'] == 1]['total']
]
 fig, axes = plt.subplots(1, 2, figsize=(18, 8))
axes = axes.flatten()
 for i, j in zip(lis_text, axes):
    try:
        new = i.str.split()
        new = new.values.tolist()
```

```
            corpus = [word.lower() for i in new for word in i]
            dic = defaultdict(int)
            for word in corpus:
                if word in stop:
                    dic[word] += 1
         #   print(dic)
            top = sorted(dic.items(), key=lambda x: x[1], reverse=True)[:15]
            #   print(top)
            x, y = zip(*top)
            df = pd.DataFrame([x, y]).T
            df = df.rename(columns={0: 'Stopword', 1: 'Count'})
            sns.barplot(x='Count', y='Stopword', data=df, palette='plasma', ax=j)
            plt.tight_layout()
        except:
            plt.close()
            print('No stopwords left in texts.')
            break

from collections import Counter, defaultdict

fig, axes = plt.subplots(1, 2, figsize=(18, 8))
axes = axes.flatten()

for i, j in zip(lis_text, axes):

    new = i.str.split()
    new = new.values.tolist()
    corpus = [word for i in new for word in i]

    counter = Counter(corpus)
    most = counter.most_common()
    x, y = [], []
    for word, count in most[:30]:
        if (word not in stop):
            x.append(word)
            y.append(count)
    sns.barplot(x=y, y=x, palette='plasma', ax=j)
print(x, y)
axes[0].set_title('Reliable')
axes[1].set_title('Unreliable')
axes[0].set_xlabel('Count')
axes[0].set_ylabel('Word')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Word')
```

```
fig.suptitle('Most Common Unigrams in Text', fontsize=24, va='baseline')
plt.tight_layout()

fig, axes = plt.subplots(1, 2, figsize=(18, 8))
axes = axes.flatten()

for i, j in zip(lis_title, axes):

    new = i.str.split()
    new = new.values.tolist()
    corpus = [word for i in new for word in i]

    counter = Counter(corpus)
    most = counter.most_common()
    x, y = [], []
    for word, count in most[:30]:
        if (word not in stop):
            x.append(word)
            y.append(count)
    sns.barplot(x=y, y=x, palette='plasma', ax=j)
print(x, y)
axes[0].set_title('Reliable')
axes[1].set_title('Unreliable')
axes[0].set_xlabel('Count')
axes[0].set_ylabel('Word')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Word')

fig.suptitle('Most Common Unigrams in Title', fontsize=24, va='baseline')
plt.tight_layout()

from sklearn.feature_extraction.text import CountVectorizer
def ngrams(n, title, lis_type):
    """A Function to plot most common ngrams"""
    fig, axes = plt.subplots(1, 2, figsize=(18, 8))
    axes = axes.flatten()
    for i, j in zip(lis_type, axes):

        new = i.str.split()
        new = new.values.tolist()
        corpus = [word for i in new for word in i]

        def _get_top_ngram(corpus, n=None):
```

```
            #getting top ngrams
            vec = CountVectorizer(ngram_range=(n, n),
                        max_df=0.9,
                        stop_words='english').fit(corpus)
            bag_of_words = vec.transform(corpus)
            sum_words = bag_of_words.sum(axis=0)
            words_freq = [(word, sum_words[0, idx])
                    for word, idx in vec.vocabulary_.items()]
            words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
            return words_freq[:15]

        top_n_bigrams = _get_top_ngram(i, n)[:15]
        x, y = map(list, zip(*top_n_bigrams))
        sns.barplot(x=y, y=x, palette='plasma', ax=j)
            axes[0].set_title('Reliable')
        axes[1].set_title('Unreliable')
        axes[0].set_xlabel('Count')
        axes[0].set_ylabel('Words')
        axes[1].set_xlabel('Count')
        axes[1].set_ylabel('Words')
        fig.suptitle(title, fontsize=24, va='baseline')
        plt.tight_layout()

ngrams(2, 'Most Common Bigrams', lis_text)
ngrams(2, 'Most Common Bigrams', lis_title)
train.head()

# Obtain the total words present in the dataset
list_of_words = []
for i in train.total:
    for j in i:
        list_of_words.append(j)

len(list_of_words)

# Obtain the total number of unique words
total_words = len(list(set(list_of_words)))
total_words
# dataframe information
train.info()
# check for null values
train.isnull().sum()
train = train[['total','label']]
train.head()
```

6. Count Vectorizer Demo

```
from sklearn.feature_extraction.text import CountVectorizer
train_set = {"the sky is blue", "the sun is bright"}
count_vec = CountVectorizer(max_features = 3)
a = count_vec.fit_transform(train_set)
a.todense()
count_vec.get_feature_names()
```

7. Applying NLP Techniques

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

X_train = train['total']
Y_train = train['label']
```

1. Bag-of-words / CountVectorizer

```
corpus = [
   'This is the first document.',
   'This document is the second document.',
   'And this is the third one.',
   'Is this the first document?',
]

# in sklearn, we can import Bag of Words model through CountVectorizer. This
works pretty much like Bag of words
# it is modified version of Bag of Words. It replaces the vector, instead of 1, w the
frequency...

# we r making the object, vectorizer, of class/module CountVectorizer()
vectorizer = CountVectorizer()

X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
print("The above words are the unique words and consists of the feature set")

print("The below matrix will show the frequency of the features in the feature set",
end="\n\n")
X.todense()
```

2. TF-iDF Vectorizer

```
def vectorize_text(features, max_features):
   vectorizer = TfidfVectorizer( stop_words='english', #it will remove the englsh
stopwords
```

```
                    decode_error='strict',
                    analyzer='word',
                    ngram_range=(1, 2), #single_words or 2words(bi-grams)
                    max_features=max_features
                    #max_df=0.5 # Verwendet im ML-Kurs unter Preprocessing
                    )
    feature_vec = vectorizer.fit_transform(features)
    return feature_vec.toarray()

tfidf_features = vectorize_text(['hello how are you doing','hi i am doing fine'],30)
# 30 here is the number of max_features
 print("creates some weight for all these words: ", end='\n\n')
tfidf_features
```

8. Applying Feature Extraction using count vectorization and tfidf

```
#Feature extraction using count vectorization and tfidf.
count_vectorizer = CountVectorizer()
count_vectorizer.fit_transform(X_train)
freq_term_matrix = count_vectorizer.transform(X_train)
tfidf = TfidfTransformer(norm="l2")
tfidf.fit(freq_term_matrix)
tf_idf_matrix = tfidf.fit_transform(freq_term_matrix)

print("10 feature names are:", end = '\n\n')
count_vectorizer.get_feature_names()[9000:9010]
tf_idf_matrix
#split in samples
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(tf_idf_matrix, Y_train,
random_state=0)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

9. Logistic Regression

```
logreg = LogisticRegression(C=1e5, random_state=110, max_iter=300)
logreg.fit(X_train, y_train)
pred = logreg.predict(X_test)
print('Accuracy of Logistic Regression on test set: {:.5f}'
    .format(logreg.score(X_test, y_test)))
print(pred)

from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,pred))
```

```
cm = confusion_matrix(y_test, pred)
ax = plt.subplot()
sns.set(font_scale=3.0) # Adjust to fit
sns.heatmap(cm, annot=True, ax=ax, cmap="coolwarm", fmt="g");

# Labels, title and ticks
label_font = {'size':'18'}  # Adjust to fit
ax.set_xlabel('Predicted labels', fontdict=label_font);
ax.set_ylabel('True labels', fontdict=label_font);

title_font = {'size':'21'}  # Adjust to fit
ax.set_title('Confusion Matrix for Logistic Regression on Fake News Dataset',
fontdict=title_font);

ax.tick_params(axis='both', which='major', labelsize=10)  # Adjust to fit
ax.xaxis.set_ticklabels(['Real', 'Fake']);
ax.yaxis.set_ticklabels(['Real', 'Fake']);
plt.show()
plt.savefig('Logistic.png')
```

10. MultinomialNB

```
NB = MultinomialNB()
NB.fit(X_train, y_train)
pred_NB = NB.predict(X_test)
print('Accuracy of MultinomialNB classifier on test set: {:.2f}'
    .format(NB.score(X_test, y_test)))

from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,pred_NB))

cm = confusion_matrix(y_test, pred_NB)
ax = plt.subplot()
sns.set(font_scale=3.0) # Adjust to fit
sns.heatmap(cm, annot=True, ax=ax, cmap="bwr", fmt="g");

# Labels, title and ticks
label_font = {'size':'18'}  # Adjust to fit
ax.set_xlabel('Predicted labels', fontdict=label_font);
ax.set_ylabel('True labels', fontdict=label_font);

title_font = {'size':'21'}  # Adjust to fit
ax.set_title('Confusion Matrix for Multinomial Naive Bayes Classifier on Fake News
Dataset', fontdict=title_font);
```

```
ax.tick_params(axis='both', which='major', labelsize=10)  # Adjust to fit
ax.xaxis.set_ticklabels(['Real', 'Fake']);
ax.yaxis.set_ticklabels(['Real', 'Fake']);
plt.show()
plt.savefig('MultinomialNB.png')
```

11. Decision Tree

```
DT = DecisionTreeClassifier()
DT.fit(X_train, y_train)
pred_dt = DT.predict(X_test)
DT.score(X_test, y_test)

print(classification_report(y_test,pred_dt))

cm = confusion_matrix(y_test, pred_dt)
ax = plt.subplot()
sns.set(font_scale=3.0) # Adjust to fit
sns.heatmap(cm, annot=True, ax=ax, cmap="seismic", fmt="g");

# Labels, title and ticks
label_font = {'size':'18'}  # Adjust to fit
ax.set_xlabel('Predicted labels', fontdict=label_font);
ax.set_ylabel('True labels', fontdict=label_font);
title_font = {'size':'21'}  # Adjust to fit
ax.set_title('Confusion Matrix for Decision Tree Classifier on Fake News Dataset',
fontdict=title_font);
ax.tick_params(axis='both', which='major', labelsize=10)  # Adjust to fit
ax.xaxis.set_ticklabels(['Real', 'Fake']);
ax.yaxis.set_ticklabels(['Real', 'Fake']);
plt.show()
plt.savefig('DT.png')
```

12. Random Forest Classifier

```
RFC = RandomForestClassifier(n_estimators=50, criterion="entropy")
RFC.fit(X_train, y_train)
pred_RFC = RFC.predict(X_test)

print(RFC.score(X_test, y_test))
print(classification_report(y_test, pred_RFC))

cm = confusion_matrix(y_test, pred_RFC)
ax = plt.subplot()
sns.set(font_scale=3.0) # Adjust to fit
sns.heatmap(cm, annot=True, ax=ax, cmap="PuOr", fmt="g");
```

```
# Labels, title and ticks
label_font = {'size':'18'}  # Adjust to fit
ax.set_xlabel('Predicted labels', fontdict=label_font);
ax.set_ylabel('True labels', fontdict=label_font);
title_font = {'size':'21'}  # Adjust to fit
ax.set_title('Confusion Matrix for Random Forest Classifier on Fake News Dataset',
fontdict=title_font);
ax.tick_params(axis='both', which='major', labelsize=10)  # Adjust to fit
ax.xaxis.set_ticklabels(['Real', 'Fake']);
ax.yaxis.set_ticklabels(['Real', 'Fake']);
plt.show()
plt.savefig('RandomForestClassifier.png')
```

13. KNN

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
pred_knn = knn.predict(X_test)

print(knn.score(X_test, y_test))
print(classification_report(y_test, pred_knn))
cm = confusion_matrix(y_test, pred_knn)
ax = plt.subplot()
sns.set(font_scale=3.0) # Adjust to fit
sns.heatmap(cm, annot=True, ax=ax, cmap="Pastel1", fmt="g");

# Labels, title and ticks
label_font = {'size':'18'}  # Adjust to fit
ax.set_xlabel('Predicted labels', fontdict=label_font);
ax.set_ylabel('True labels', fontdict=label_font);
title_font = {'size':'21'}  # Adjust to fit
ax.set_title('Confusion Matrix for K-Nearest-Neighbours Classifier on Fake News
Dataset', fontdict=title_font);
 ax.tick_params(axis='both', which='major', labelsize=10)  # Adjust to fit
ax.xaxis.set_ticklabels(['Real', 'Fake']);
ax.yaxis.set_ticklabels(['Real', 'Fake']);
plt.show()
plt.savefig('K-Nearest-Neighbours.png')
```

14. SVM -Linear Kernel

```
svm_ = svm.SVC(kernel="linear")
svm_.fit(X_train, y_train)
pred_svm = svm_.predict(X_test)
print(svm_.score(X_test, y_test))
print(classification_report(y_test, pred_svm))
```

```
 cm = confusion_matrix(y_test, pred_svm)
ax = plt.subplot()
sns.set(font_scale=3.0) # Adjust to fit
sns.heatmap(cm, annot=True, ax=ax, cmap="Paired", fmt="g");

# Labels, title and ticks
label_font = {'size':'18'}  # Adjust to fit
ax.set_xlabel('Predicted labels', fontdict=label_font);
ax.set_ylabel('True labels', fontdict=label_font);
 title_font = {'size':'21'}  # Adjust to fit
ax.set_title('Confusion Matrix for SVM Classifier on Fake News Dataset',
fontdict=title_font);
 ax.tick_params(axis='both', which='major', labelsize=10)  # Adjust to fit
ax.xaxis.set_ticklabels(['Real', 'Fake']);
ax.yaxis.set_ticklabels(['Real', 'Fake']);
plt.show()
plt.savefig('SVM.png')
```