**FAKE NEWS DETECTION USING NLP**

## PROJECT REPORT

***Submitted by***

# Kanchan Singh (1801341015)

***in partial fulfilment for the award of the degree of***

# BACHELOR OF TECHNOLOGY

***in***

## COMPUTER SCIENCE AND ENGINEERING

***Under the guidance of***

# Asst Prof. Satyabrat Sahoo



## SILICON INSTITUTE OF TECHNOLOGY

**SAMBALPUR, ODISHA, 768200**



**FAKE NEWS DETECTION USING NLP**

## PROJECT REPORT

***Submitted by***

# Kanchan Singh (1801341015)

***in partial fulfilment for the award of the degree of***

# BACHELOR OF TECHNOLOGY

***in***

## COMPUTER SCIENCE AND ENGINEERING

***Under the guidance of***

# Asst Prof. Satyabrat Sahoo



## SILICON INSTITUTE OF TECHNOLOGY

**SAMBALPUR, ODISHA, 768200**



**BIJU PATNAIK UNIVERSITY OF TECHNOLOGY: ORISSA**



# BONAFIDE CERTIFICATE

### This is to certify that this Project Report entitled “Fake News Detection using NLP” is the bonafide work done and submitted by Kanchan Singh bearing Regd.no: 1801341015 in partial fulfilment of the requirement for the award of B.Tech. in Computer Science and Engineering of SILICON INSTITUTE OF TECHNOLOGY SAMBALPUR during the academic session 2021-2022.

|  |  |
| --- | --- |
| **Mr. Satyabrat Sahoo GUIDE** | **Mr. Satyabrat Sahoo HEAD OF DEPARTMENT**  **CSE** |

**EXTERNAL**

# ACKNOWLEDGEMNT

I would like to give a special gratitude to my Project guide, Mr. Satyabrat Sahoo Computer Science & Engineering, whose contribution in simulating suggestions and encouragement helped me to coordinate my project especially in writing this report. I am greatly indebted to him for providing his valuable guidance at all stages of the study, his advice, constructive suggestions, positive and supportive attitude and continuous encouragement which helped me a lot during my learning process.

I take this opportunity to express my sincere thanks to Mr. Satyabrat Sahoo, Head of the Department, Computer Science & Engineering for providing the necessary facilities in the department.

Furthermore, I would also like to acknowledge with much appreciation the critical role of my parents and friends for encouraging and helping me complete my project.

Kanchan Singh (1801341015)

# DECLARATION

I declare that this written submission represents my ideas. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academics honestly and integrity and have not misrepresented any idea in my submission. I understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Kanchan Singh (1801341015)

Table of Contents

[List of Figures… i](#_TOC_250067)

Abbreviation and Acronyms… ii

[Abstract… iii](#_TOC_250066)

[Chapter 1 Introduction 1](#_TOC_250065)

* 1. [Fake News 1](#_TOC_250064)
  2. [Relevance of the Project… 2](#_TOC_250063)
  3. [Problem Statement… 3](#_TOC_250062)
  4. [Objective 3](#_TOC_250061)

[Chapter 2 Literature Review 4](#_TOC_250060)

* 1. [Introduction 4](#_TOC_250059)
  2. A Review of Fake News Detection Methods using ML 5
  3. Fake News Detection Using Machine Learning Algorithm 5
  4. [Fake News Detection Using Logistic Regression… 6](#_TOC_250058)
  5. Comparison between Multinomial and

Bernoulli Naïve Bayes for Text Classification 6

* 1. [Fake News Detection on social media: A Data Mining Perspective 7](#_TOC_250057)

Chapter 3 Basics of Machine Learning 8

* 1. [Machine Learning… 8](#_TOC_250056)
     1. [Features of Machine Learning… 9](#_TOC_250055)
  2. [Classification of Machine Learning… 9](#_TOC_250054)
     1. [Supervised Learning… 9](#_TOC_250053)
     2. [Unsupervised Learning… 10](#_TOC_250052)
     3. [Reinforcement Learning… 10](#_TOC_250051)
  3. [Machine Learning Life Cycle… 10](#_TOC_250050)
  4. [Natural Language Processing (NLP)… 11](#_TOC_250049)
     1. [Stages in NLP 12](#_TOC_250048)

[Chapter 4 Project Description 13](#_TOC_250047)

* 1. [Existing System… 13](#_TOC_250046)
  2. [Proposed System… 13](#_TOC_250045)
  3. [Feasibility Study… 14](#_TOC_250044)
     1. [Economic Feasibility… 14](#_TOC_250043)
     2. [Technical Feasibility… 14](#_TOC_250042)
     3. [Social Feasibility… 14](#_TOC_250041)
  4. [System Specification… 14](#_TOC_250040)
     1. [Hardware Specification… 15](#_TOC_250039)
     2. [Software Specification… 15](#_TOC_250038)
     3. [Standards and Policies… 16](#_TOC_250037)

Chapter 5 System Analysis and Design 17

* 1. [General Architecture 17](#_TOC_250036)
     1. [Text Collection… 18](#_TOC_250035)
     2. [Text Preprocessing… 18](#_TOC_250034)
     3. [Feature Extraction… 18](#_TOC_250033)
     4. Classifiers… 19
  2. [Design Phase 23](#_TOC_250032)
     1. [Data flow Diagram 23](#_TOC_250031)
     2. [Process flow Diagram… 24](#_TOC_250030)
     3. [Deployment flow Diagram… 25](#_TOC_250029)
     4. [Collaboration Diagram… 25](#_TOC_250028)
     5. [Sequence Diagram 26](#_TOC_250027)
  3. [Module Description 26](#_TOC_250026)

[Chapter 6 Implementing and Testing… 27](#_TOC_250025)

* 1. [Input and Output… 27](#_TOC_250024)
     1. [Input Design… 27](#_TOC_250023)
     2. [Output Design… 27](#_TOC_250022)
  2. [Model Construction… 27](#_TOC_250021)
  3. [The Web Interface 39](#_TOC_250020)
  4. [Common Platform: Flask… 45](#_TOC_250019)
  5. [Testing 46](#_TOC_250018)
  6. [Types of Testing 46](#_TOC_250017)
     1. [Unit Testing 46](#_TOC_250016)
     2. [Integration Testing 47](#_TOC_250015)
     3. [Functional Testing 47](#_TOC_250014)
     4. [Test Result 48](#_TOC_250013)
  7. Testing Strategy 49

Chapter 7 Results and Discussion 50

* 1. [Efficiency of the Proposed System 50](#_TOC_250012)
  2. [Comparison of Existing and Proposed System 50](#_TOC_250011)
  3. [Advantages of the Proposed System 51](#_TOC_250010)
  4. [Results 51](#_TOC_250009)
     1. [Confusion Matrix 51](#_TOC_250008)
     2. [Accuracy 51](#_TOC_250007)
     3. [Precision 51](#_TOC_250006)
     4. [Recall 52](#_TOC_250005)
     5. [F1 – Score 52](#_TOC_250004)
  5. [Model Deployment 52](#_TOC_250003)

Chapter 8 Conclusion and Future Enhancement… 54

* 1. [Conclusion… 54](#_TOC_250002)
  2. [Future Enhancements… 54](#_TOC_250001)

[References… 55](#_TOC_250000)

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Figure Numbers** | **Page No.** |
| 1.1 | Growth of Fake news in 2016 presidential election | 2 |
| 3.1 | Graphical representation of relationship between various fields in AI | 9 |
| 3.2 | Machine Learning life cycle | 10 |
| 5.1 | Architecture Diagram | 17 |
| 5.2 | Logistic Regression | 20 |
| 5.3 | Data Flow Diagram | 23 |
| 5.4 | Process Flow | 24 |
| 5.5 | Deployment flow | 25 |
| 5.6 | Collaboration Diagram | 25 |
| 5.7 | Sequence diagram | 26 |
| 6.1 | True News dataset | 29 |
| 6.2 | Fake News dataset | 29 |
| 6.3 | Dataset before pre-processing | 30 |
| 6.4 | Dataset after pre-processing | 30 |
| 6.5 | True News dataset Word-Cloud | 31 |
| 6.6 | Fake News dataset Word-Cloud | 31 |
| 6.7 | Logistic Regression Output | 32 |
| 6.8 | Decision Tree Output | 33 |
| 6.9 | Random Forest Output | 34 |
| 6.10 | Stochastic Gradient Descent output | 34 |
| 6.11 | Gradient Descent Boosting output | 35 |
| 6.12 | XGBOOST output | 36 |
| 6.13 | Multinomial Naïve Bayes output | 37 |
| 6.14 | Bernoulli Naïve Bayes output | 37 |
| 6.15 | Proposed Model output | 39 |
| 6.16 | Unit Testing | 46 |
| 6.17 | Integration Testing | 47 |
| 6.18 | Functional Testing | 48 |
| 6.19 | Test Result | 48 |
| 7.1 | Confusion Matrix | 51 |
| 7.2 | Output page for True news | 53 |
| 7.3 | Output page for Fake news | 53 |

**ABBREVIATIONS AND ACRONYMS**

|  |  |
| --- | --- |
| NLP | Natural Language Processing |
| NLTK | Natural Language Toolkit |
| TFIDF | Term frequency and Inverse Document Frequency |
| EDA | Exploratory Data Analysis |
| SGD | Stochastic Gradient Descent |
| POS | Parts OF Speech |

# ABSTRACT

The effect of fraud news has increased exponentially within the recent past and something must be done to stop this from continuing within the future. Fake news is a current trend going on with the emergence of technology. It provides an opportunity to spread false news that will hurt the common people. It is associated with the practice of spreading false and/or misleading information so as to influence public opinion on the news. This practice is known as disinformation. It is one among the most weapons utilized in distracting countries in Information warfare, which is listed as an emerging Cybersecurity threat. In this we explore fake news as a disinformation tool. Many techniques are surveyed for the detection process of fake news. The term fake news has become a buzz word lately. There was a time when if anyone needed any news, he or she would await the next-day newspaper. However, now a days with the increase of newspapers, news channels people can get fast updates of the news despite of whether it is true or false. People must be careful when confirming about the news by proper resource. Nowadays social-networking systems, online news portals, and other online media became the most sources of stories through which interesting and breaking news are shared at a rapid pace. However, many news portals serve interest by feeding with distorted, partially correct, and sometimes imaginary news that’s likely to draw in the eye of a target group of individuals. It is a major concern now for people whether to believe the news or not. This confusion will mislead the people in many ways. The Aim of this project is to use the Natural Language Processing and Machine learning to detect the Fake news based on the text content of the Article. After building the suitable Machine learning model to detect the fake/true news then to deploy it into a web interface using python Flask.

**Keywords: Fake news, Natural language processing, Web interface, Flask**

# CHAPTER 1 INTRODUCTION

Data or information is the most valuable asset. The most important problem to be solved is to evaluate whether the data is relevant or irrelevant. Fake data has a huge impact on lot of people and organizations. Since fake news tends to spread fast than the real news there a need to classify news as fake or not. In the project the dataset used is from Kaggle website where real news and fake news are in two separate datasets. We combined both the datasets into one and trained with different machine learning classification algorithms to classify the news as fake or not. In this project different feature engineering methods for text data has been used which is going to convert the text data into feature vectors which is sent into machine learning algorithms to classify the news as fake or not.

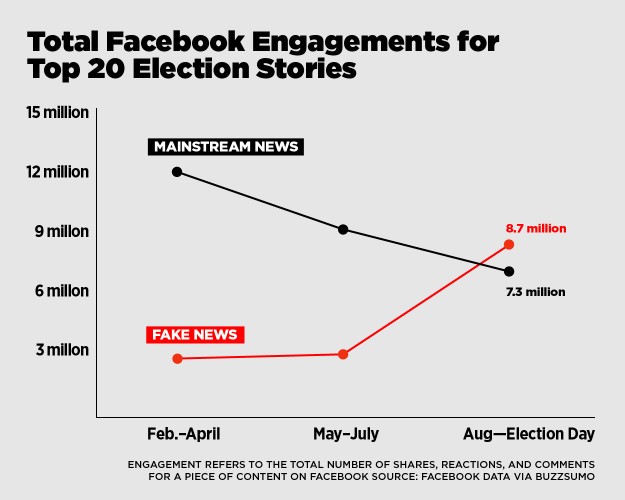
With different features and classification algorithms we are going to classify the news as fake or real and the algorithm with the feature which gives us the best result with feature extraction method and algorithm. We are going to predict the news as fake or real. In this project we will be ignoring attributes like the source of the news, whether it was reported online or in print, etc. and instead focus only the content matter being reported. We aim to use different machine learning algorithms and determine the best way to classify news.

## Fake News

Fake news is false or misleading information presented as [news](https://en.wikipedia.org/wiki/News). Fake news often has the aim of damaging the reputation of a person or entity, or making money through [advertising](https://en.wikipedia.org/wiki/Advertising) revenue. The prevalence of fake news has increased with the recent rise of [social media,](https://en.wikipedia.org/wiki/Social_media) especially the [Facebook News Feed,](https://en.wikipedia.org/wiki/Facebook_News_Feed) and this misinformation is gradually seeping into the mainstream media. Several factors have been implicated in the spread of Fake news, such as political polarization, post-truth politics, motivated reasoning, confirmation bias, and social media algorithms.

During the 2016 US presidential election, various kinds of fake news about the candidates widely spread in the online social networks, which may have a significant effect on the election results. According to a post-election statistical report, online social networks account for more than 41.8% of the fake news data traffic in the election, which is much greater than the data traffic shares of both traditional TV/radio/print medium and online search engines respectively.

Fake news detection is becoming increasingly difficult because people who have ill intentions are writing the fake pieces so convincingly that it is difficult to separate from real news.



**Fig 1.1 Growth of Fake news in 2016 presidential election**

Multiple strategies for fighting fake news are currently being actively researched, for various types of fake news. Politicians in certain autocratic and democratic countries have demanded effective self-regulation and legally-enforced regulation in varying forms, of social media and web search engines.

## Relevance of the project

A fake news classification system using different feature extraction methods and different classification algorithms like Logistic Regression, Stochastic gradient descent, XG-BOOST, Decision Tree, Random Forest, Bernoulli Naïve Bayes, Multinomial Naïve Bayes and the best algorithms chosen we are going to use it in predicting the news as fake or real.

In order to create a real time application, the algorithm should be fed with the most recent data. Data is of different sizes so that should be properly cleaned to get better results. So, we are using different algorithms and feature extraction methods to get the best result.

## Problem Statement

Our main aim of the project is to make a machine learning model, with the help of which news can be classified as fake or real with help of different machine learning classification algorithms, natural language processing toolkits, ensemble model techniques and text feature extraction methods for classifying news.

## Objective

To achieve our goal of developing machine learning model to classify news as fake or real, we need perform following tasks in the same order as stated.

* + - Data Collection and Analysis
    - Pre-processing the data
    - Text feature extraction
    - Using different classification algorithms
    - Taking the best classification algorithms
    - Implementing Voting Classifier mechanism
    - Classifying the news as fake or real.
    - Deploying the model.

# CHAPTER 2 LITERATURE REVIEW

* 1. **Introduction**

In Today's world, anybody can post the content over the internet. Unfortunately, counterfeit news gathers a lot of consideration over the web, particularly via web-based networking media. Individuals get misdirected and don't reconsider before flowing such mis-educational pieces to the most distant part of the arrangement. Such type of activities are not good for the society where some rumours or vague news evaporates the negative thought among the people or specific category of people. As fast the technology is moving, on the same pace the preventive measures are required to deal with such activities. Broad communications assuming a gigantic job in impacting the general public and as it is normal, a few people attempt to exploit it. There are numerous sites which give false data. They deliberately attempt to bring out purposeful publicity, deceptions and falsehood under the pretence of being true news. Their basic role is to control the data that can cause open to have confidence in it. There are loads of case of such sites everywhere throughout the world. Therefore, counterfeit news influences the brains of the individuals. As indicated by study Scientist accept that numerous man-made brainpower calculations can help in uncovering the bogus news. Fake news detection is made to stop the rumours that are being spread through the various platforms whether it be social media or messaging platforms, this is done to stop spreading fake news which leads to activities like mob lynching, this has been a great reason motivating us to work on this project. We have been continuously seeing various news of mob lynching that leads to the murder of an individual; fake news detection works on the objective of detecting this fake news and stopping activities like this thereby protecting the society from these unwanted acts of violence. The digital news industry in the United States is facing a complex future. On one hand, a steadily growing portion of Americans are getting news through the internet, many U.S. adults get news on social media, and employment at digital-native outlets has increased. On the other, digital news has not been immune to issues affecting the broader media environment, including layoffs, made- up news and public distrust. Half of Americans (52%) say they have changed the way they use social media because of the issue of made-up news. Furthermore, among the Americans who ever get news through social media, half have stopped following a news source because they thought it was posting made-up news and information. At the same time, about a third (31%)

of social media news consumers say they at least sometimes click on news stories they think are made up. So, there is need to stop the fake news spreading

* 1. **A Review of Fake News Detection Methods using Machine Learning**

This article describes, a survey on the state of art pertaining to the type of fake news and solutions that are being proposed. The research in this field has been going on for a long time and in the Indian context, the ill effects of spreading fake news are far from what anyone might think. Unlike in the context of other countries, WhatsApp is the prime distributor of fake news as compared to other social networking sites like Facebook and Twitter. Due to the increase of internet users in India, which has increased 137million (in 2012) to over 600 million (in 2019) facing unique challenges day by day.

This survey deals with a review of existing machine learning algorithms Naïve Bayes, Convolutional Neural Network, LSTM, Neural Network, Support Vector Machine proposed for detecting and reducing fake news from different social media platforms like Facebook, WhatsApp, twitter, etc. This review provides a comprehensive detail including data mining perspective, evaluation metrics, and representative datasheets. Further, a comparison of the state-of-the-art is presented and the untackled challenges in detecting fake news are highlighted. Research in the field of detecting fake news has been hampered to a great extent by the lack of quantity and quality of existing datasets. Therefore, this review compares the existing approaches to build models and with further improvements to be expected by using the combination of different machine learning techniques.

## Fake News Detection Using Machine Learning Algorithms

The goal of the research is to perform binary classification of various news articles available online with the help of concepts pertaining to Artificial Intelligence, Natural Language Processing and Machine Learning. We aim to provide the user with the ability to classify the news as fake or real and also check the authenticity of the website publishing the news.

This paper explains the system which is developed in three parts. The first part is static which works on machine learning classifier. The model with 4 different classifiers is trained, studied and the best classifier is chosen for final execution. The second part is dynamic which takes the keyword/text from user and searches online for the truth probability of the news. The third part provides the authenticity of the URL input by user. In this paper, Python and its Sci-kit

libraries is used. Python has a huge set of libraries and extensions, which can be easily used in Machine Learning. Sci-Kit Learn library is the best source for machine learning algorithms where nearly all types of machine learning algorithms are readily available for Python, thus easy and quick evaluation of ML algorithms is possible.

## FAKE NEWS DETECTION USING LOGISTIC REGRESSION

This article describes a simple fake news detection method based on one of the artificial intelligence algorithms – Logistic Regression. The goal of the research is to examine how this particular method works for this particular problem given a manually labelled news dataset and to support (or not) the idea of using artificial intelligence for fake news detection. The difference between these article and articles on the similar topics is that in this paper Logistic Regression was specifically used for fake news detection; also, the developed system was tested on a relatively new data set, which gave an opportunity to evaluate its performance on a recent data.

In machine learning, Logistic Regression is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

A detection model for fake news using TF-IDF analysis through the lenses of different feature extraction techniques has been created. Different feature extraction and machine learning techniques were investigated. The proposed model achieved an accuracy of approximately 72% when using TF-IDF features and logistic regression classifier

## Comparison between Multinomial and Bernoulli Naïve Bayes for Text Classification

The paper aims to predict that whether the sentiment of the news article is positive or negative using the two popular approaches of Naïve Bayes Text Categorization i.e., Multivariate Bernoulli Naïve Bayes Classification and Multinomial Naïve Bayes Classification. Also, the research aims to identify that which approach between the given two approaches perform better for the given dataset.

Multinomial Naïve Bayes classifier works on the concept of term frequency which means that how many times does the word occur in a document. This model tells two facts that whether the word occur in a document or not as well as its frequency in that document. While predicting the polarity of a new news article, we multiply the probabilities of the occurrence of all the words in the article against both the polarities and the one which is higher gives the polarity of this article

On the other hand, Bernoulli Naïve Bayes Classifier works on the binary concept that whether the term occurs in a document or not but unlike Multinomial Naïve Bayes, it does not tell about the term frequency. While predicting the polarity of a new news article, we multiply the probabilities of the occurrence of all the words in the article and also the probabilities of non- occurrence of words which do not occur in the article against both the polarities and the one which is higher gives the polarity of this article.

This paper aims to use both of these algorithms for classification of textual news articles on many important events that happened in India in 2018.

## Fake News Detection on social media: A Data Mining Perspective

In this survey, a comprehensive review of detecting fake news on social media, including fake news characterizations on psychology and social theories, existing algorithms from a data mining perspective, evaluation metrics and representative datasets is presented.

In this paper, the conceptual characterization of traditional fake news and fake news in social media is done. Based on this characterization, the problem definition and proposed approaches for fake news detection are explored.

# CHAPTER 3

**BASICS OF MACHINE LEARNING**

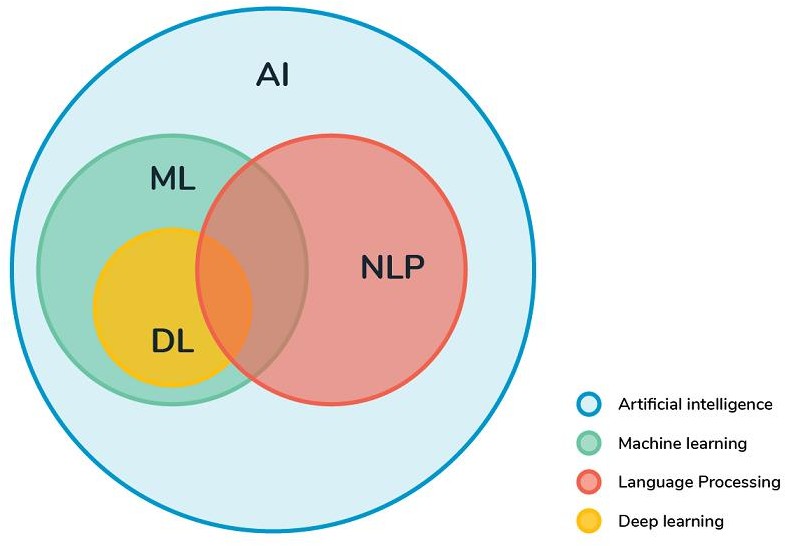
## Machine Learning

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.” This is Alan Turing’s definition of machine learning.

Deep learning is a class of machine learning algorithms that utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach. The word "deep" in "deep learning" refers to the number of layers through which the data is transformed. More precisely, deep learning systems have a substantial credit assignment path (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feedforward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized). For recurrent

neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited. Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design,

medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts.



**Fig. 3.1 Graphical representation of relationship between various fields in AI**

## Features of Machine Learning:

* + - * Machine learning uses data to detect various patterns in a given dataset.
      * It can learn from past data and improve automatically.
      * It is a data-driven technology.
      * Machine learning is much similar to data mining as it also deals with the huge amount of the data.

## Classification of Machine Learning

Machine learning can be classified into three types:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

## Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labelled data to the machine learning system in order to train it, and on that basis, it predicts the output.

The example of supervised learning is spam filtering. Supervised learning can be grouped further in two categories of algorithms:

* + - * Classification
      * Regression

## Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision. It can be further classifieds into two categories of algorithms:

#### Clustering

* + - * **Association**

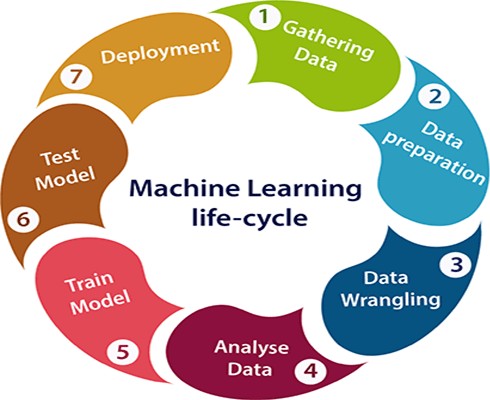
## Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

## Machine learning Life cycle

Machine learning life cycle involves seven major steps, which are given below:



**Fig 3.2 Machine Learning life cycle**

* Gathering Data - The goal of this step is to identify and obtain all data-related problems.
* Data preparation - Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.
* Data Wrangling - It is the process of cleaning and converting raw data into a useable format. collected data may have various issues, including:
  + Missing Values
  + Duplicate data
  + Invalid data
  + Noise

So, we use various filtering techniques to clean the data.

* Analyse Data - The aim of this step is to build a machine learning model to analyse the data using various analytical techniques and review the outcome.

This step involves:

* + Selection of analytical techniques
  + Building model
  + Review the result.
* Train the model
* Test the model
* Deployment

## Natural Language Processing (NLP)

NLP is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to fruitfully process large amounts of natural language data.

Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyse large amounts of natural language data.

## Stages in NLP

#### LEXICAL ANALYSIS

Lexical Analysis involves identifying and the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of txt into paragraphs, sentences, and words.

#### SYNTACTIC ANALYSIS (PARSING)

Syntactic Analysis involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as “The school goes to boy” is rejected by English syntactic analyser.

#### SEMANTIC ANALYSIS

Semantic Analysis draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyser disregards sentence such as “hot ice- cream”.

#### DISCOURSE INTEGRATION

The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence. So, in Discourse Integration gives the meaning based on all the sentences given before it. E.g., Consider the sentence “Water is flowing on the bank of the river” But bank has two meanings One Financial Institute and Two River of the bank here System has to consider the second meaning.

#### PRAGMATIC ANALYSIS

During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

# CHAPTER 4 PROJECT DESCRIPTION

## Existing System

There is a lot of research happening on the machine learning models on detecting the fraud most of it focusing on online site reviews and social media posts only. Particularly after the 2016 American presidential elections the research on fake news algorithms is a question mark to the subject of particular attention. Many Researchers proposed several approaches that aims in giving perfect results in classifying miss leading articles. They find normal n-grams method and parts of speech techniques are failing in important context classification. These methods won’t be enough rather than these methods are combined with some other methods to deep analysis the content. Deep analysis includes powerful algorithms, context free grammar using n-grams techniques to produce a better result. Existing systems implemented on semantic analysis at object descriptor pairs with the contradictions on other method like deep syntax technique for additional advantage. Also, other structures developed with vector space model with the same success rate.

## Proposed System

Here, the model is build using many classifiers used to detect fake news. All the data is fed into the classifiers. Since the main important thing is text classification so using naive bayes approach gives best results as it is best test processing technique. The main goal is to choose which text processing approach and which text need to be used. Next step is to find optimal features for training the model usually selected after clear removal of stop words, comas, unnecessary special symbols etc. Once fitting with all the features then accuracy scores and confusion matrix is compared. As the number of approaches increased, many techniques are implemented to extract best working model with high accuracy score gained by training with 61,000+ records which gives correct prediction on the articles.

## Advantages

* + - Easy to understand and implement
    - Provides Good accuracy scores

## Feasibility Study

The best performing models got 87.04 accuracy score. This is due to dataset still containing some stop words. Because by using 61,000+ records it is so difficult for the model to train the data. The model is not implemented with deep learning techniques, because 87 percent accuracy with more data is not bad. By clear analysing logistic regression was the best approach for classifying the fake news. Logistic regression classifier is saved as model.pkl file as it is will be used in web interface deployed using flask server. The text field takes the input from the user and gives back the output in the same web interface like Fake news or true news.

## Economic Feasibility

The context of fake news information, fake news consumers, creators, various arbiters reinforced and formed as a vicious circle and developed cost benefit structure of engaging in this activity. Mostly fake news detection needs technical approach so many companies can create their cost-efficient detection system which boosts their own business into profits.

## Technical Feasibility

All the machine learning classifiers ideas implemented in this project, and used classifiers which gives good and accurate predictions of many news articles and 6 the hardware requirements support every business organization. The model implemented in a way, that uses only best prediction classifier to predict the output.

## Social Feasibility

The model is deployed using flask with a good UI design. So, people can interact easily with the web interface without many confusions. The user can enter the text in the textbox provided and they will get the prediction in less than a minute. Fake news detector aimed at analysing and identifying false articles so the scale of any business organisation before launching any product gives them a good social impact.

## System Specification

The service model gives the prediction less than 10 seconds. So, the normal computer with less specifications is enough to get the output. The project requires that the machine has python 3.9 installed in it. After installing python, need to setup PATH variables in edit system configuration.

## Hardware Specification

* GPUs providing the 2–4x faster speed of CPU clock speed
* Range of 3500 (GPU) vs 16 (CPU).
* A CPU as i5–7500U can train an average of 100 examples/second
* Power consumption is around 200 W to 250 and also needs a PC that additionally require 100 W of power, which leads to a total of 300W.

## Software Specification

Some packages need to be downloaded and installed in the system along with python, anaconda navigator or any other tools like visual studio code, PyCharm etc

* + - * SK-Learn (scikit-learn)
      * NumPy 7
      * Pandas
      * Matplotlib
      * Seaborn
      * NLTK
      * Job-Lib
      * flask

#### To install the required python packages

* + - * pip install -U scikit-learn
      * pip install NumPy
      * pip install Pandas
      * pip install matplotlib
      * pip install Seaborn
      * pip install nltk
      * pip install flask
      * pip install joblib

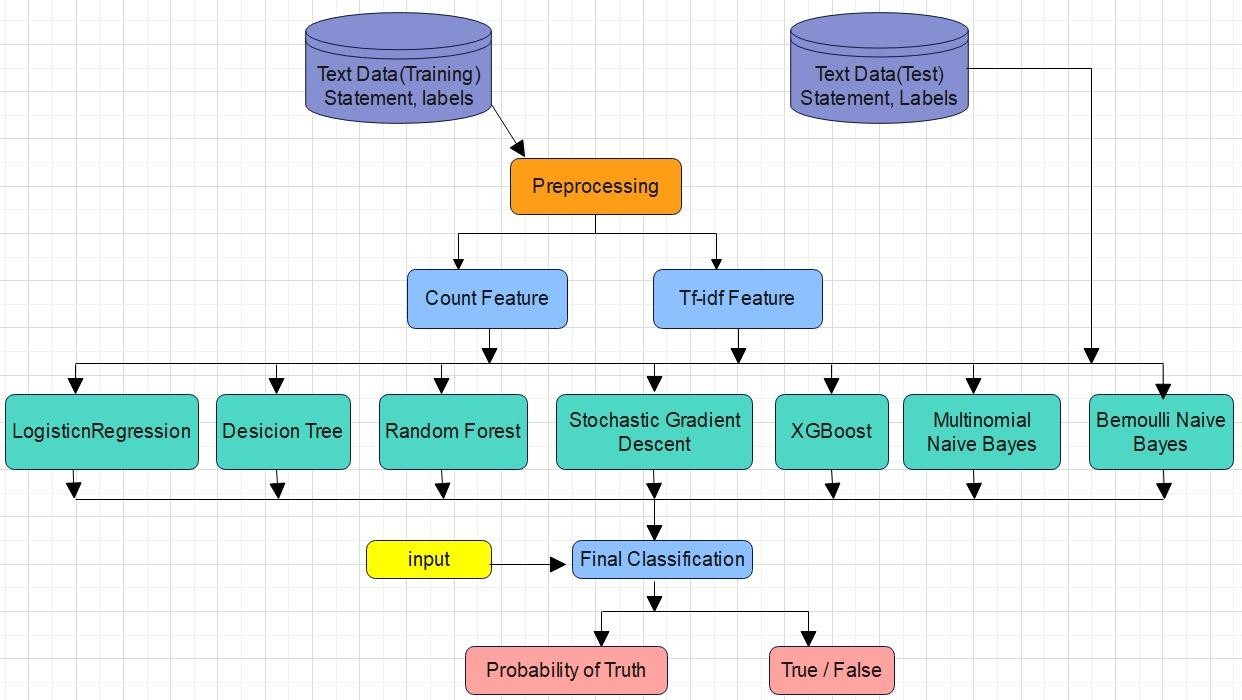
## Standards and Policies

* + - * Peripheral Devices - Standards used: ISO 2382-12:1993
      * Jupyter notebook- Standards used: ISO 3166-1:2018
      * Anaconda- Standards used: ISO 8061-11:2017
      * Python- Standard used: ISO6160:1979

# CHAPTER 5

**SYSTEM ANALYSIS AND DESIGN**

## General Architecture



**Fig 5.1 Architecture Diagram**

Generally, dataset inputs are classified into train and test data. Test data is kept aside and useful for testing. Training data is pre-processed using count features and TF-IDF features. After pre- processing all the stop words are removed. All the train data is fed into various machine learning algorithms to predict output.

## Text Collection

The dataset was taken from Kaggle the content and metadata has been extracted from 244 web sites that have been considered to be associated with fake news by the BS Detector Chrome Extension by Daniel Sieradski. It consists of almost 13000 posts over a period of 30 days. Research on this dataset using language processing tools has already been carried out by Kaggle users. The dataset was generated by Andrew Thompson to create document term matrices using the articles and analyse connections between articles using common political affiliations, medium or subject matter. It contains articles from top 15 American publications

and the articles were mostly published between the years of 2016 and 2017. It consists of around 150000 articles that were collected by scraping news website homepages and RSS feeds. However, we will randomly select only 13000 articles from this dataset and merge it with the fake news dataset for more accurate predictions and for avoiding a skewed dataset.

## Text Pre-processing

After a text is obtained, we start with text pre-processing. Text pre-processing includes: Converting all letters to lower case

* + - * Removing numbers
      * Removing punctuations, accent marks
      * Removing white spaces Removing stop words

## Feature Extraction

Text needs to be converted into numbers before it is used with a machine learning algorithm. For classification of documents, documents are taken as input and a class label is generated as output by the predictive algorithm. The documents need to be converted into fixed-length vectors of numbers for the algorithm to take them as input. The input for the machine learning algorithm are the words encoded as integers or float point values.

## Bag of Words (BOW)

We make the list of unique words in the text corpus called vocabulary. Then we can represent each sentence or document as a vector with each word represented as 1 for present and 0 for absent from the vocabulary.

## Count Vectorizer

Count Vectorizer generates an encoded vector that contains the length of the entire vocabulary coupled with the frequency of each word by which it appears in the document.

## Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency (TF) = (Number of times term t appears in a document) / (Number of terms in the document)

𝑡𝑓𝑖,𝑗

= 𝑛𝑖,𝑗 ……………………………………………………………5.1

∑ 𝑛𝑖,𝑗

𝑛

𝑘=0

Inverse Document Frequency (IDF) = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in. The IDF of a rare word is high, whereas the IDF of a frequent word is likely to be low. Thus, having the effect of highlighting words that are distinct.

𝑖𝑑𝑓(𝑤) = log ( 𝑁 ) 5.2

𝑑𝑓𝑡

We calculate TF-IDF value of a term as = TF \* IDF.

𝑤𝑖,𝑗

= 𝑡𝑓𝑖,𝑗

𝑥 log ( 𝑁 ) 5.3

𝑑𝑓𝑡

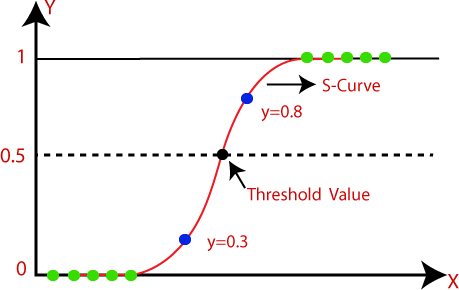
## Classifier

The feature vectors are sent to the classifier to classify the news as fake or not

## Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labelled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labelled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labelled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labelling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from logistic unit, hence the alternative names. Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the odds ratio.

The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cut-off value and classifying inputs with probability greater than the cut-off as one class, below the cut-off as the other; this is a common way to make a binary classifier.



**Fig 5.2 Logistic Regression**

## Decision Tree

Decision Tree is one of Supervised Machine Learning (output label is given) technique where the data is split consecutively through a definite parameter. In a decision tree every node speaks to a feature(attribute), each link(branch) speaks to a decision(rule) and each leaf speaks to an outcome (categorical or continuous value). The entire algorithm is to make a tree for the entire information and process a solitary result at each leaf(or limit error in each leaf). The process of text classification using decision tree marks internal node as terms and the branches withdrawing from them as derived weight, and relative class labels are represented by leaf node. Decision tree utilizes query structure throughout the path of the tree classifying the document from root until it reaches a definite leaf node. In memory decision tree construction, majority of training data will not fit and results inefficient due to swapping the training tuples. This is dealt in as FDT to deal with the multiclass record which lessens the induction cost. A symbolic rule induction system based on decision tree is presented to improve text classification to implement multiclass classification.

## Random Forest

As the name suggests, Random Forest algorithm generates the forest with a number of decision trees. So, it is the collection of decision trees. Decision trees are attractive classifiers among others because of their high execution speed. Based on random samples from the database a random forest classifier averages multiple decision trees. Generally, the more trees in the forest, is the sign that forest is robust. Similarly in the random forest classifier, high accuracy is obtained by higher the number of trees in the forest. While concurrently creating a tree with decision nodes, a decision tree breaks the dataset down into smaller subsets. The decision root node is selected through highest information gain and leaf nodes based on a pure subset for each iteration simultaneously. Calculation of Information Gain (IG) requires impurity measure (Entropy) of that node. There are various indices to measure the degree of impurity. A leaf node represents a category or pure subset. The trees in a random forest are created under random data so there might be chances to be lack meaning and noisy. In order to make a model with low variance random forest averages these trees. The irrelevant trees drop each other out and the staying meaningful trees yield the final result.

## Stochastic Gradient Descent

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high dimensional optimization problems this reduces the very high computational burden, achieving faster iterations in trade for a lower convergence rate.

* + - 1. **Gradient Boosting**

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor’s error. In contrast to AdaBoost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

## XG-BOOST

This is an ensemble method that seeks to create a strong classifier (model) based on “weak” classifiers. In this context, weak and strong refer to a measure of how correlated are the learners to the actual target variable. By adding models on top of each other iteratively, the errors of the previous model are corrected by the next predictor, until the training data is accurately predicted or reproduced by the model. Among these classifiers and feature extraction method the best classification algorithm and feature extraction method is used to classify the news as fake or real

## Multinomial Naïve Bayes

The multinomial naïve Bayes is widely used for assigning documents to classes based on the statistical analysis of their contents. It provides an alternative to the "heavy" AI-based semantic analysis and drastically simplifies textual data classification.

The classification aims to assign fragments of text (i.e., documents) to classes by determining the probability that a document belongs to the class of other documents, having the same subject.

Each document consists of multiple words (i.e., terms), that contribute to an understanding of a document’s contents. A class is a tag of one or multiple documents, referring to the same subject. The labelling of documents with one of the existing classes is done by performing the statistical analysis, testing the hypothesis that a document’s terms already occurred in other documents from a particular class. This increases the probability that a document is from the same class as the documents, already classified.

* + - 1. **Bernoulli Naïve Bayes**

Bernoulli Naive Bayes is one of the variants of the Naive Bayes algorithm in machine learning. It is very useful to be used when the dataset is in a binary distribution where the output label is present or absent.

In the multivariate Bernoulli event model, features are independent Booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence features are used rather than term frequencies. If xi is a boolean expressing the occurrence or absence of the *i*'th term from the vocabulary, then the likelihood of a document given a class Ck is given by :

𝑷(𝑿⁄𝑪

) = ∏𝒏

𝒙𝒊

(𝟏−𝒙𝒊)………………………5.4

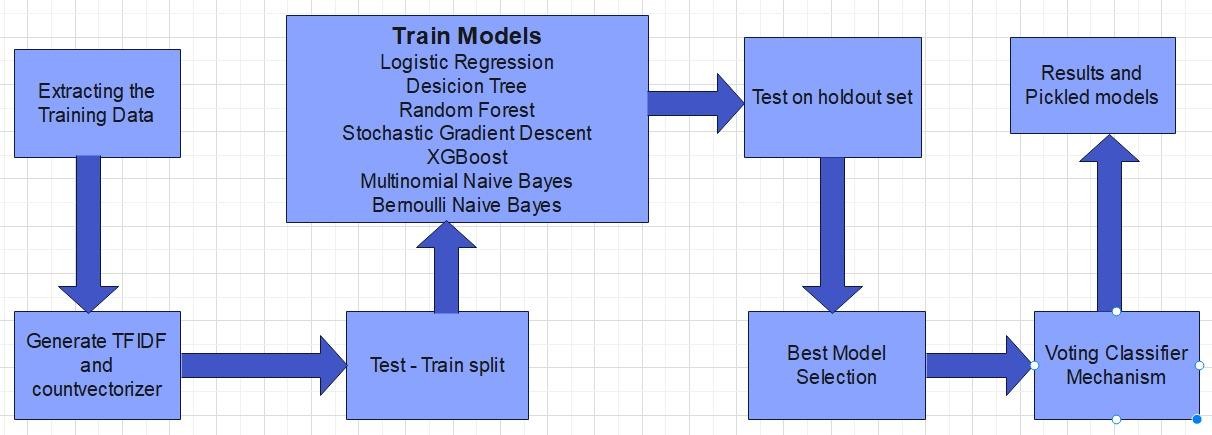
𝒌 𝒊=𝟏 𝒑𝒌𝒊(𝟏 − 𝒑𝒌𝒊)

where pki is the probability of class Ck generating the term xI. This event model is especially popular for classifying short texts. It has the benefit of explicitly modelling the absence of terms. Note that a naive Bayes classifier with a Bernoulli event model is not the same as a multinomial NB classifier with frequency counts truncated to one.

## Design Phase

## Data Flow Diagram

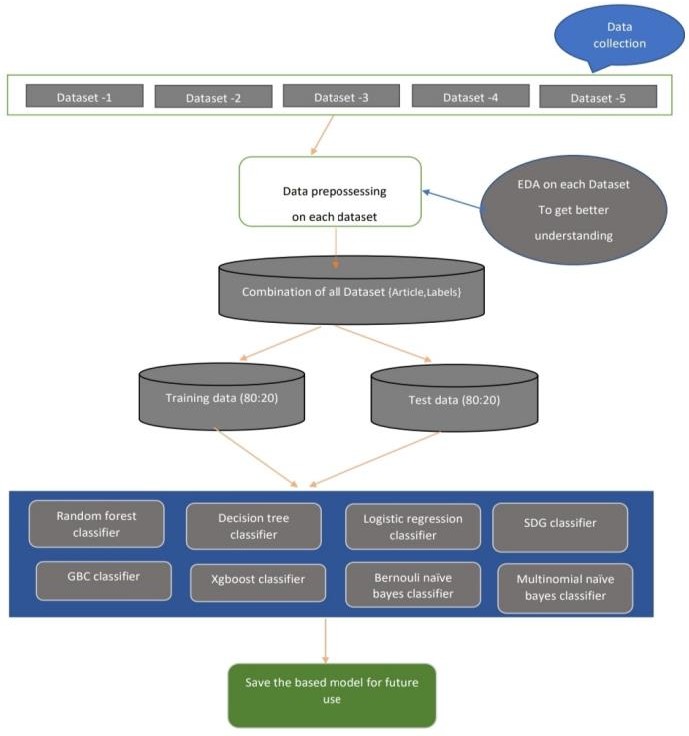
It shows how data is processed in terms of input and output. Input data is again split into test and train data after generating TFIDF and count-vectorizer. Test/Train split is fed into algorithms. The predicted output is compared with the actually stored test data with different algorithms. Final algorithm is chosen to store as pickle model.



**Fig 5.3 Data Flow Diagram**

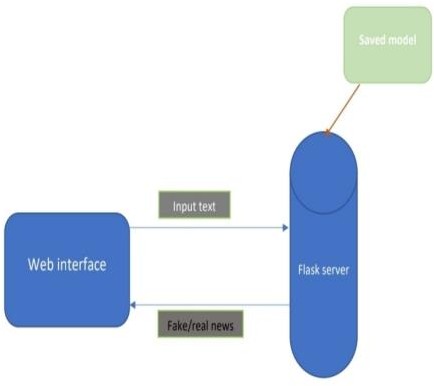
## Process flow Diagram

Five datasets are used for generating this model. Next phase is data Pre-processing here EDA techniques are implemented to get clear understanding of the features used. Later they are split as train and test data. Machine learning algorithms like Random Forest, decision tree, Logistic Regression, SDG classifier etc gives accuracy scores. By that we store better classifier for future use.



**Fig 5.4 Process Flow**

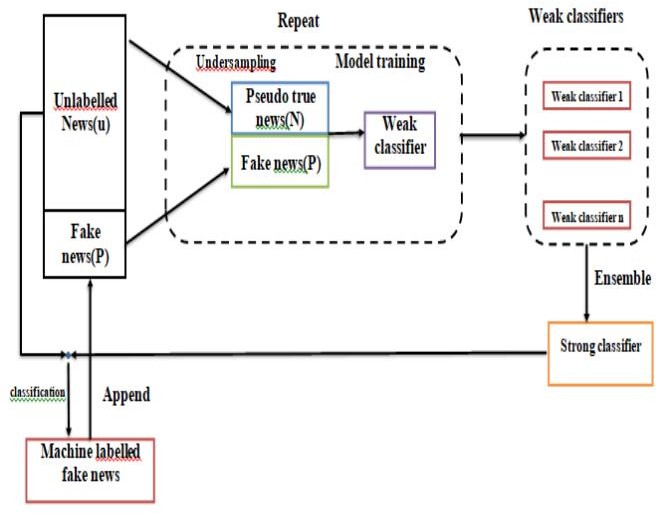
## Deployment flow Diagram



**Fig 5.5 Deployment flow**

Out of 7, 3 best are chosen and voting mechanism is applied to get the desired model and it is saved as model.pkl and it is used by the flask server displaying the output. A Web interface is created containing text field, submit button and output display functionality. Whenever user types the text in the text field, flask server renders the output from the stored pickle model. It fetches the output as true or false and displays the same at the bottom the screen.

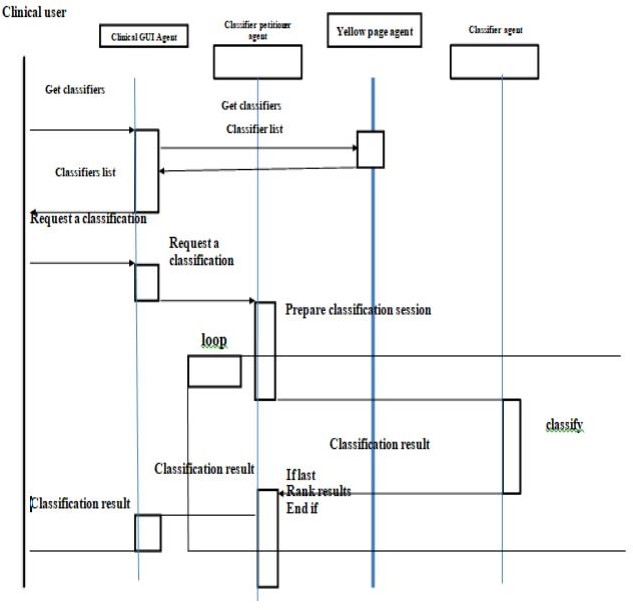
## Collaboration Diagram



**Fig 5.6 Collaboration Diagram**

## Sequence Diagram

Model is provided with all the useful classifiers. Training data is set up as epochs. Each time some set of data are trained with classifiers and generate an output. It works as a loop until all the data available is trained.



**Fig 5.7 Sequence diagram**

#### Module Description

The above diagrams show clear understanding of how the model works from basic building blocks to the end. Different diagrams show the approaches used while building the model.

# CHAPTER 6 IMPLEMENTING AND TESTING

## Input and Output

Data sets collected from kaggle.com are available as a public domain. Different datasets contain different information and different columns. For final model building we don’t need all the fields available. So important features are extracted from it. Data pre-processing is done for all the data.

## Input Design

Input datasets contains many unwanted fields like articles, tweets, id’s, Headlines etc. For model building only text and label field is required. So, the final dataset taken contains only two columns [‘Article’, ‘Label’]. New column article is created for text which is combination of header and text. Five datasets are used and more than 61000+ records are chosen for the model building. In the label columns two binary numbers ’0’ and ’1’ is used. 0 represents fake news and 1 represents true news. Finally, input is trained with different classifiers.

## Output Design

Model is deployed using Flask. Voting classifier results saved in model.pkl file. This file is used for the output prediction. A web interface is designed with text field and button. For the input news provided by the user in the text field the output will be displayed as true news or fake news with red and green color indication of it.

## Model Construction

#### Importing important packages

import nltk import re import string

import numpy as np import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer from sklearn.metrics import accuracy\_score, confusion\_matrix from sklearn.feature\_extraction.text import CountVectorizer

from sklearn import feature\_extraction, linear\_model, model\_selection, preprocessing from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.pipeline import Pipeline,FeatureUnion from sklearn.naive\_bayes import BernoulliNB from sklearn.naive\_bayes import MultinomialNB from sklearn.tree import DecisionTreeClassifier from sklearn.linear\_model import SGDClassifier

from sklearn.ensemble import RandomForestClassifier,VotingClassifier import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression from wordcloud import WordCloud, STOPWORDS

from sklearn.metrics import (accuracy\_score, f1\_score, precision\_score, recall\_score)

from sklearn.calibration import CalibratedClassifierCV from sklearn.svm import LinearSVC,SVC

* **Data Collection**

There are 5 different datasets taken and data analysis and preprocessing is done on each of them

Dataset2\_true = pd.read\_csv("True.csv") Dataset2\_fake = pd.read\_csv("Fake.csv")

Dataset2\_true.nunique()

title 20826

text 21192

subject 2

date 716 dtype: int64

#Counting by Subjects in Real news

for key,count in Dataset2\_true.subject.value\_counts().iteritems(): print(f"{key}:\t{count}")

#Getting Total Rows

print(f"Total Records:\t{Dataset2\_true.shape[0]}")

politicsNews: 11272

worldnews: 10145

Total Records: 21417

#Counting by Subjects in Fake news

for key,count in Dataset2\_fake.subject.value\_counts().iteritems(): print(f"{key}:\t{count}")

#Getting Total Rows

print(f"Total Records:\t{Dataset2\_fake.shape[0]}")

News: 9050

politics: 6841

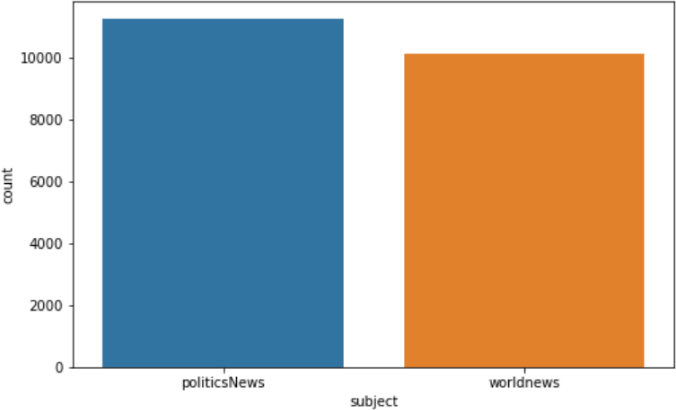
left-news: 4459

Government News: 1570

US\_News: 783

Middle-east: 778

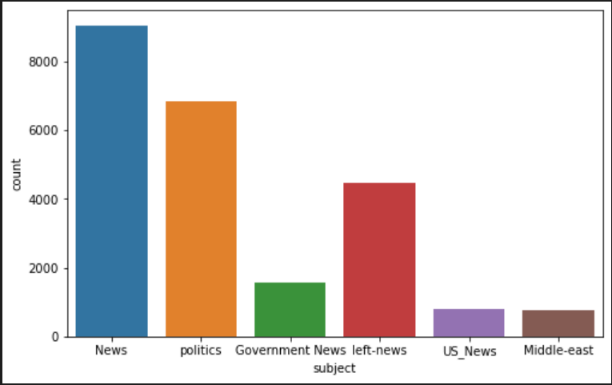
Total Records: 23481



#ploting the Subjects in Real news plt.figure(figsize=(8,5)) sns.countplot("subject", data=Dataset2\_true) plt.show()

**Fig 6.1 True News dataset**

#ploting the Subjects in Fake news plt.figure(figsize=(8,5)) sns.countplot("subject", data=Dataset2\_fake) plt.show()



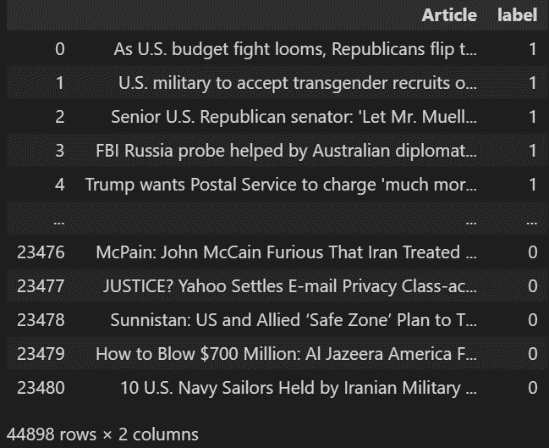
**Fig 6.2 Fake News dataset**

Dataset2\_true['label']= 1

Dataset2\_fake['label']= 0

Dataset2 = pd.concat([Dataset2\_true, Dataset2\_fake]) Dataset2["Article"] = Dataset2["title"] + Dataset2["text"] Dataset2.sample(frac = 1) #Shuffle 100%

Dataset2 = Dataset2.loc[:,['Article','label']] Dataset2



#### Data Preprocessing

**Fig 6.3 Dataset before preprocessing**

In this step we will clean the data that will be used for training. The cleaning will involve these steps -:

Removing all the extra information like brackets, any kinds of punctuations – commas, apostropes, quotes, question marks, and more.

Remove all the numeric text, urls.

def wordpre(text): text = text.lower()

text = re.sub('\[.\*?\]', '', text)

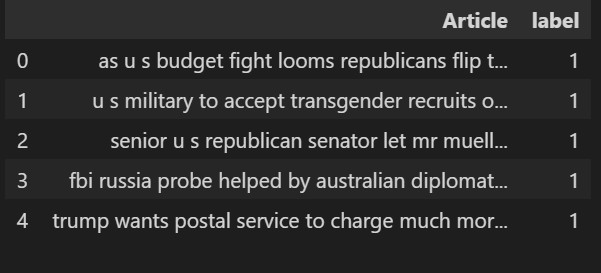
text = re.sub("\\W"," ",text) # remove special chars text = re.sub('https?://\S+|www\.\S+', '', text)

text = re.sub('<.\*?>+', '', text)

text = re.sub('[%s]' % re.escape(string.punctuation), '', text) text = re.sub('\n', '', text)

text = re.sub('\w\*\d\w\*', '', text)

return text



## Applying the wordpre method to the dataset

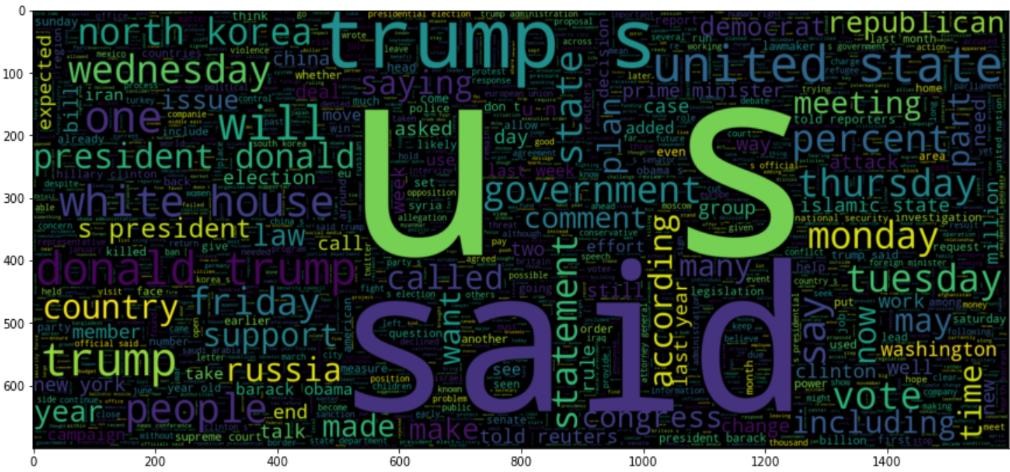
Dataset2['Article']=Dataset2['Article'].apply(wordpre) Dataset2.head()

**Fig 6.4 Dataset after Preprocessing**

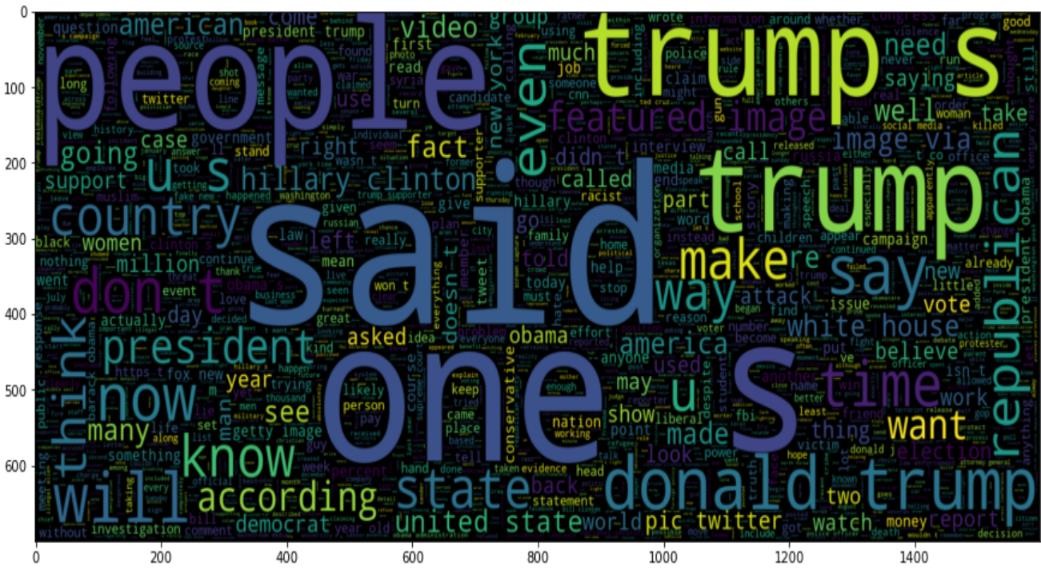
#word used in Real news

plt.figure(figsize=(15,15)) wc=WordCloud(max\_words=2000, width=1600, height=700,

stopwords=STOPWORDS).generate("".join(Dataset2[Dataset2.label== 1].Article)) plt.imshow(wc, interpolation="bilinear")



**Fig 6.5 True News dataset wordcloud**



#word used in Fake news plt.figure(figsize=(15,15))

wc=WordCloud(max\_words=2000, width=1600, height=700, stopwords=STOPWORDS).generate("".join(Dataset2[Dataset2.label== 0].Article)) plt.imshow(wc, interpolation="bilinear")

**Fig 6.6 Fake News dataset wordcloud**

In similar way all the datasets are preprocessed and analysed.

#### Train /Test Data

x\_train,x\_test,y\_train,y\_test = train\_test\_split(Dataset['Article'], Dataset['label'], test\_size=0.2, random\_state=2020)

x\_train.shape

(61184,)

x\_test.shape

(15196,)

#### Feature Extraction and Training with different classification Algorithms Logistic Regression

#LogisticRegression

lr\_pipe = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('model', LogisticRegression())])

Logisticmodel = lr\_pipe.fit(x\_train, y\_train) prediction = Logisticmodel.predict(x\_test)

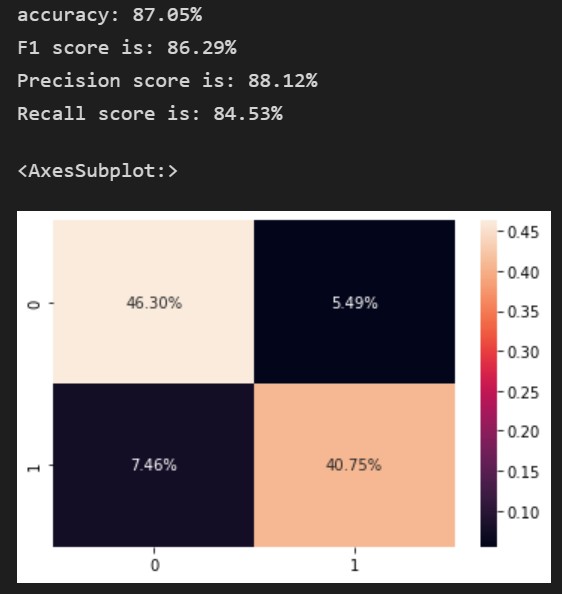
print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) Logisticmodel\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2)

f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')



**Fig 6.7 Logistic Regression Output**

#### Decision Tree

#####DecisionTreeClassifier

dtc\_pipe = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),

('model', DecisionTreeClassifier(criterion= 'entropy',

max\_depth = 10, splitter='best',random\_state=2020))])

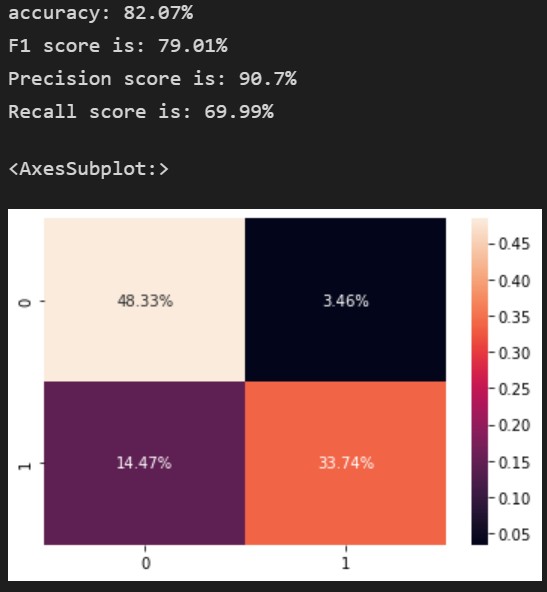
DecisionTreemodel = dtc\_pipe.fit(x\_train, y\_train) prediction = DecisionTreemodel.predict(x\_test)

print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) DecisionTreemodel\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2) f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')



**Fig 6.8 Decision Tree Output**

#### Random Forest

#####RandomForestClassifier

pipe = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),

('model', RandomForestClassifier())])

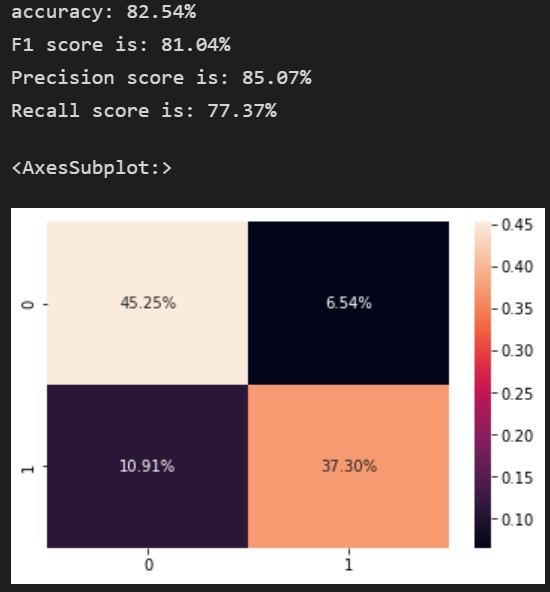
RandomForestmodel = pipe.fit(x\_train, y\_train) prediction = RandomForestmodel.predict(x\_test)

print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) RandomForestmodel\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2) f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

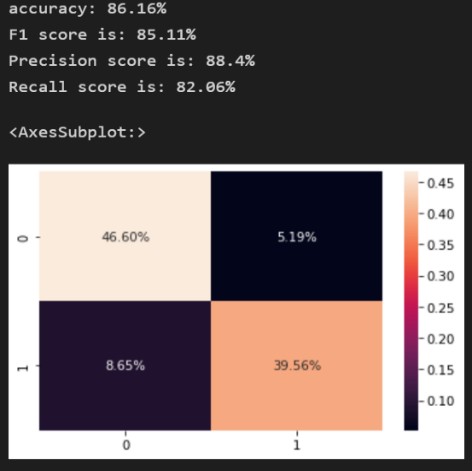
print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')



**Fig 6.9 Random Forest Output**

#### Stochastic Gradient Descent



#Stochastic Gradient Descent

sgdpipe = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('model', SGDClassifier())])

SGDmodel = sgdpipe.fit(x\_train, y\_train) prediction = SGDmodel.predict(x\_test)

print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) SDGmodel\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2)

f1 = f1\_score(y\_test, prediction)

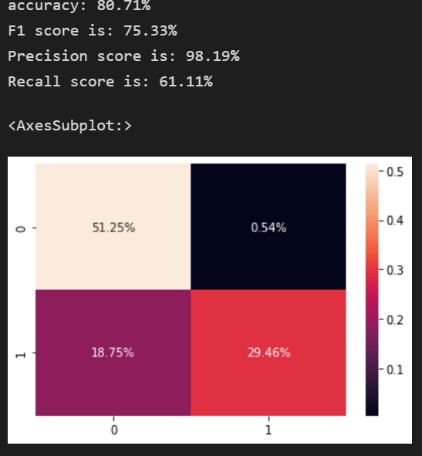
prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')

**Fig 6.10 Stochastic Gradient Descent output**

#### Gradient Boosting



#GradientBoostingClassifier

from sklearn.ensemble import GradientBoostingClassifier gbcpipe = Pipeline([('vect', CountVectorizer()),

('tfidf', TfidfTransformer()),

('model', GradientBoostingClassifier(loss = 'deviance',

learning\_rate = 0.01,

n\_estimators = 10,

max\_depth = 5, random\_state=2020))])

GBCmodel = gbcpipe.fit(x\_train, y\_train) prediction = GBCmodel.predict(x\_test)

print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) GBCmodel\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2)

f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')

**Fig 6.11 Gradient Descent Boosting output**

#### XGBOOST

#########XGBClassifier

from xgboost import XGBClassifier

xgpipe = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),

('model', XGBClassifier(loss = 'deviance',

learning\_rate = 0.01,

n\_estimators = 10,

max\_depth = 5,

random\_state=2020))])

xgboostmodel = xgpipe.fit(x\_train, y\_train) prediction = xgboostmodel.predict(x\_test)

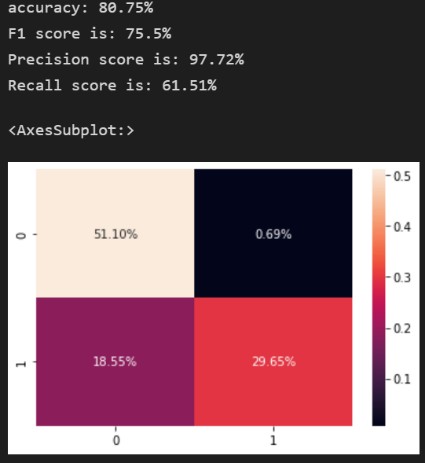
print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) xgboostmodel\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2)

f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')



**Fig 6.12 XGBoost output**

#### Multinomial Naïve Bayes

#######Multinomial Naive Bayes Classifier pipe = Pipeline([('vect', CountVectorizer()),

('tfidf', TfidfTransformer()), ('model', MultinomialNB())])

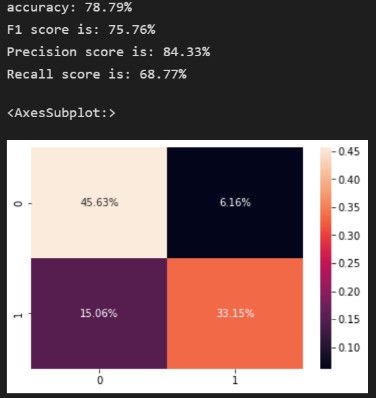
MNBCmodel = pipe.fit(x\_train, y\_train) prediction = MNBCmodel.predict(x\_test)

print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) Multinomial\_Naive\_Bayes\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2) f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

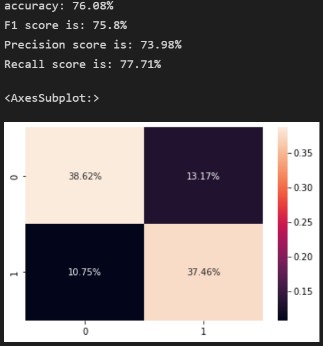
print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')



**Fig 6.13 Multinomial Naïve Bayes output**

#### Bernoulli Naïve Bayes



#############Bernoulli Naive Bayes Classifier pipe = Pipeline([('vect', CountVectorizer()),

('tfidf', TfidfTransformer()), ('model', BernoulliNB())])

BNBCmodel = pipe.fit(x\_train, y\_train) prediction = BNBCmodel.predict(x\_test)

print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) Bernoulli\_Naive\_Bayes\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2) f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')

**Fig 6.14 Bernoulli Naïve Bayes output**

#### Proposed Algorithm

###################

sgd\_voting\_pipeline = Pipeline([('sgd\_union', FeatureUnion([ ('tfidf', Pipeline([

('TF', TfidfVectorizer(lowercase=True, ngram\_range=(1, 1), stop\_words=STOPWORDS, use\_idf=True, smooth\_idf=True))

]))

])),

('sgd\_clf', SGDClassifier(alpha=0.0001, average=False, class\_weight=None, epsilon=0.1,

eta0=0.0, fit\_intercept=True, l1\_ratio=0.15, learning\_rate='optimal', loss='hinge', n\_jobs=1,

penalty='l2', power\_t=0.5, random\_state=2020, shuffle=True, verbose=0, warm\_start=False))])

clf =sgd\_voting\_pipeline.fit(x\_train, y\_train)

calibrator = CalibratedClassifierCV(clf, cv=5, method='sigmoid') ###############

lr\_voting\_pipeline = Pipeline([ ('lr\_union', FeatureUnion([

('tfidf', Pipeline([

('TF', TfidfVectorizer(lowercase=True, ngram\_range=(1, 1), stop\_words=STOPWORDS, use\_idf=True, smooth\_idf=True))

]))

])),

('lr\_clf', LogisticRegression(C=0.0001, n\_jobs=-1))

]) ####################

dtc\_voting\_pipeline = Pipeline([ ('dtc\_union', FeatureUnion([

('tfidf', Pipeline([

('TF', TfidfVectorizer(lowercase=True, ngram\_range=(1, 1), stop\_words=STOPWORDS, use\_idf=True, smooth\_idf=True))

]))

])),

('dtc\_clf',DecisionTreeClassifier(criterion= 'entropy',

max\_depth = 10, splitter='best',random\_state=2020))

]) ############################

voting\_classifier = VotingClassifier(estimators=[

('lr',lr\_voting\_pipeline ),('dtc',dtc\_voting\_pipeline),('sg',calibrator)], voting='soft', n\_jobs=-1)

####Model Construction vb\_model=voting\_classifier.fit(x\_train,y\_train) prediction = vb\_model.predict(x\_test)

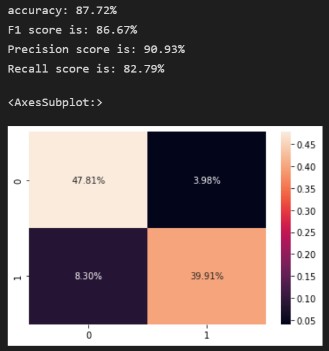
print("accuracy: {}%".format(round(accuracy\_score(y\_test, prediction)\*100,2))) votingClassifier\_accuracy = round(accuracy\_score(y\_test, prediction)\*100,2)

f1 = f1\_score(y\_test, prediction)

prec = precision\_score(y\_test, prediction) recall = recall\_score(y\_test, prediction)

print("F1 score is: {}%".format(round(f1\_score(y\_test, prediction)\*100,2))) print("Precision score is: {}%".format(round(precision\_score(y\_test, prediction)\*100,2))) print("Recall score is: {}%".format(round(recall\_score(y\_test, prediction)\*100,2))) c=confusion\_matrix(y\_test,prediction)

sns.heatmap(c/np.sum(c),annot=True,fmt='.2%')



**Fig 6.15 Proposed Model output**

## The Web Interface

This was the simplest part.

1. HTML for building the basic skeleton: HTML makes the structure of the web application and also there are some of the functions that can be achieved best with HTML only.
2. CSS for design: The CSS part is for designing only. Because it will give a more beautiful aspect to the website.

<!DOCTYPE html>

<html>

<head>

<title>Fake News Detection</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Alfa+Slab+One&display=swap');

\* {

padding: 0;

margin: 0;

box-sizing: border-box;

}

@import url('https://fonts.googleapis.com/css2?family=Pacifico&display=swap');

body{

background-image: linear- gradient(rgba(4,9,30,0.8),rgba(4,9,30,0.8)),url(https://img.freepik.com/fre e-vector/technology-background-with-earth-circuit-diagram\_1017- 19385.jpg?size=626&ext=jpg&ga=GA1.2.61866188.1654197725

);

background-size: cover; background-repeat: no-repeat;

}

.container{

display: flex;

justify-content: center; position: absolute; margin: 80px 450px;

flex-direction: column;

border: 1px solid rgb(219, 209, 209); border-radius: 20px 0px 20px 0px; background-color:#fad2a78a;

min-height: 420px; min-width: 480px;

box-shadow: 10px 10px 5px rgb(24, 18, 18);

-moz-box-shadow: 10px 10px 5px #ccc;

-webkit-box-shadow: 10px 10px 5px rgb(46, 38, 38);

-khtml-box-shadow: 10px 10px 5px #ccc;

}

.textp{

text-align: center; font-weight: bolder;

font-family: 'Pacifico', cursive; color: aqua;

}

.tarea{

font-size: larger; font-weight: bolder;

}

.waviy {

margin-top: 25px; margin-left: 10px;

position: relative;

-webkit-box-reflect: below -20px linear-gradient(transparent, rgba(0,0,0,.2));

font-size: 40px;

}

.waviy span {

font-family: 'Alfa Slab One', cursive;

position: relative; display: inline-block; color: rgb(225 11 11);; text-transform: uppercase;

animation: waviy 1s infinite; animation-delay: calc(.1s \* var(--i));

}

@keyframes waviy { 0%,40%,100% {

transform: translateY(0)

} 20% {

transform: translateY(-20px)

}

}

form{

display: flex;

flex-direction: column; align-items: center;

}

form textarea{ margin: 10px 0px;

}

form input{

width: 150px; font-size: 16px;

background-color: rgb(202, 197, 197);

}

.texts{

margin-left: 30px; margin-right: 10px; margin-top: 10px; font-weight: 500;

}

.center{

background: linear-gradient(rgba(245, 187, 105, 0.8),rgba(164, 195,

210, 0.8));

min-width: 100vw; position: absolute; text-align: center; top: 5%;

left: 50%;

transform: translate(-50%,-50%);

}

.center h1{

color: rgba(255,0,0,0.1);

font-size: 50px;

text-transform: uppercase; font-weight: 700; background-size: cover;

background-image: url(https://s3.envato.com/files/4114f57a-b4c2-4669- 8a25-c84d171b7be8/inline\_image\_preview.jpg);

-webkit-background-clip: text;

animation: background-text-animation 13s linear infinite; font-weight: bolder;

font-family: 'Pacifico', cursive;

}

@keyframes background-text-animation { 0%{

background-position: left 0px top 50%;

} 50%{

background-position: left 1500px top 50%;

} 100%{

background-position: left 0px top 50%;

}

}

.box {

height: 20vh; width: 25vw; display: flex;

justify-content: center; align-items: center; align-content: center; margin-left: 30px;

}

.text {

align-items: center; font-size: 24px;

font-family: sans-serif; color: rgb(0, 207, 90);

}

span[class^="l"] {

transition: all 0.3s ease-in-out;

}

.l1 {

animation: 2.4s letter ease-in-out infinite;

}

.l2 {

animation: 2.4s letter ease-in-out infinite; animation-delay: 0.2s;

}

.l3 {

animation: 2.4s letter ease-in-out infinite; animation-delay: 0.4s;

}

.l4 {

animation: 2.4s letter ease-in-out infinite; animation-delay: 0.6s;

}

.l5 {

animation: 2.4s letter ease-in-out infinite; animation-delay: 0.6s;

}

.l6 {

animation: 2.4s letter ease-in-out infinite; animation-delay: 0.4s;

}

.l7 {

animation: 2.4s letter ease-in-out infinite; animation-delay: 0.2s;

}

.l8 {

animation: 2.4s letter ease-in-out infinite; animation-delay: 0.2s;

}

.l9 {

animation: 2.4s letter ease-in-out infinite;

}

@keyframes letter {

0% {

font-size: 24px; padding-left: 0;

padding-right: 0;

} 50% {

font-size: 40px;

color: rgb(190, 251, 197);

padding-right: 4px; padding-left: 4px;

}

100% {

font-size: 24px; padding-left: 0;

padding-right: 0;

}

}

</style>

</head>

<body>

<div class="center">

<h1><strong>Breaking News</strong></h1>

</div>

<div class="container">

<form action="/" method="POST">

<label for="" class="tarea">Enter the news Content</label>

<textarea name="txt" rows="4" cols="50">

</textarea>

<input type="submit" name="submit">

</form>

<br/>

<div class="texts">

<h3>Enterd Text is: </h3>

{{ request.form['txt'] }}

</div>

<div class="texts">

<h2> The Ml Model Found That is a:</h2>

<p>{% if result == 0 %}

<div class="waviy">

<span style="--i:1">F</span>

<span style="--i:2">A</span>

<span style="--i:3">K</span>

<span style="--i:4">E</span>

<span style="--i:5"> </span>

<span style="--i:6">N</span>

<span style="--i:7">E</span>

<span style="--i:8">W</span>

<span style="--i:9">S</span>

</div>

{% else %}

<div class="box">

<p class="text">

<span class="l1"><strong>T</strong></span><span class="l2"><strong>R</strong></span><span

class="l3"><strong>U</strong></span><span class="l4"><strong>E</strong></span><span class="l5"><strong>

</strong></span><span class="l6"><strong>N</strong></span><span class="l7"><strong>E</strong></span><span class="l8"><strong>W</strong></span><span class="l9"><strong>S</strong></span>

</p>

</div>

{% endif %}

</p>

</div>

</div>

</body>

</html>

## Common Platform: Flask

This acts as a common platform and takes the input with the pickle module and passes it to the machine learning model afterwards the prediction is shown on the screen with the HTML and CSS website.

1. Building functions for taking input.
2. Passing input values through the ML model.
3. Using the Pickle module for serializing and de-serializing the dataset.
4. Providing output.

## App building with Flask

from flask import Flask,render\_template,url\_for,request import joblib

import re import string

import pandas as pd

app = Flask( name )

Model = joblib.load(r'C:\Users\KANCHAN SINGH\fake\_news\_detection\model.pkl')

@app.route('/') def index():

return render\_template('index.html')

def wordpre(text): text = text.lower()

text = re.sub(r'\[.\*?\]', '', text)

text = re.sub("\\W"," ",text) # remove special chars text = re.sub(r'https?://\S+|www\.\S+', '', text)

text = re.sub('<.\*?>+', '', text)

text = re.sub('[%s]' % re.escape(string.punctuation), '', text) text = re.sub('\n', '', text)

text = re.sub(r'\w\*\d\w\*', '', text) return text

@app.route('/',methods=['POST']) def pre():

if request.method == 'POST': txt = request.form['txt']

txt = wordpre(txt) txt = pd.Series(txt)

result = Model.predict(txt)

return render\_template('index.html', result = result) return ' '

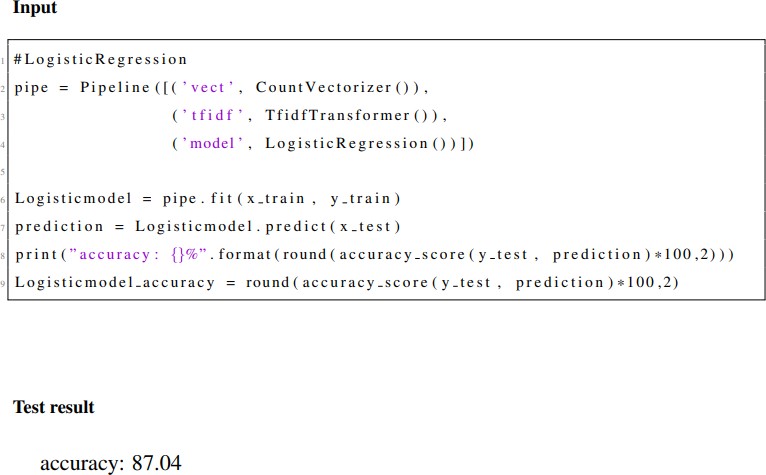
if name == " main ": app.run(debug=True)

## Testing

The main purpose of testing in machine learning is to detect false training of models and outputs. Also testing the model performance in terms of accuracy, Fl scores etc. It is different for conventional software development and machine learning model development.

## Types of Testing

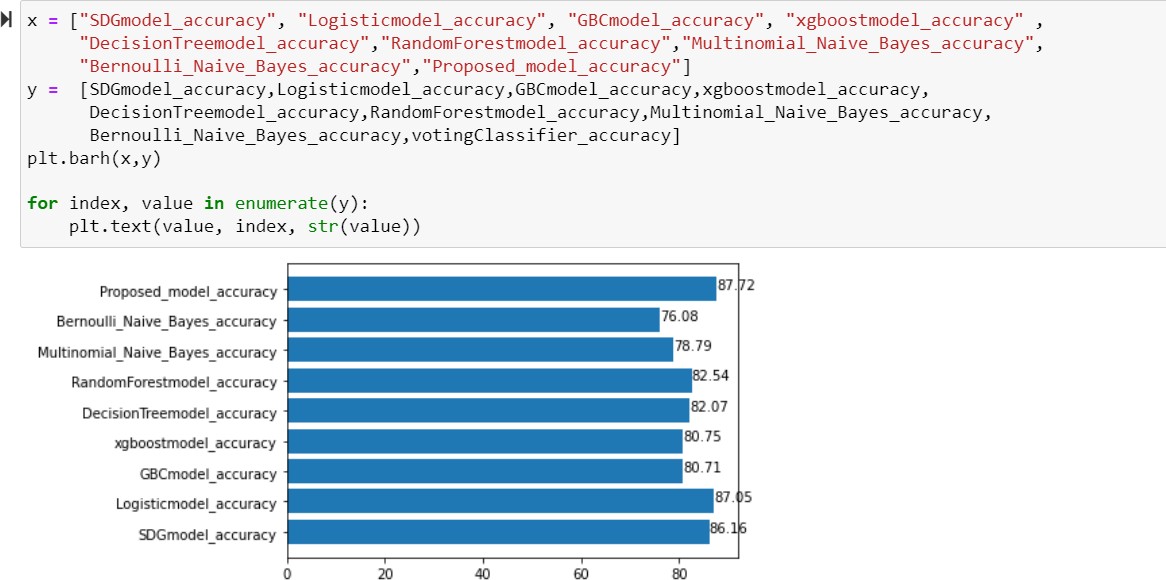
## Unit testing



**Fig 6.16 Unit Testing**

Data which we taken may or may not be perfect. So, pipelines are useful to handle this. Unit testing can check that our model gets proper data or not for training the model. Pipelining cover all our needs and it will take care about the data processing in to our model. Here in this unit testing plays a vital role. After processing the data bad data or uncleaned data may be displayed after one or more epochs.

## Integration testing

This test determines whether models generated separately when combined work properly or not. Here the model is divided by different algorithms and after K-fold cross validation

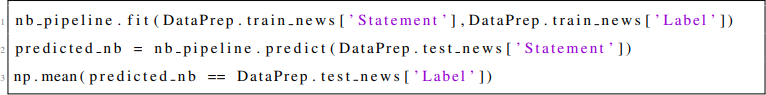
**Fig 6.17 Integration Testing**

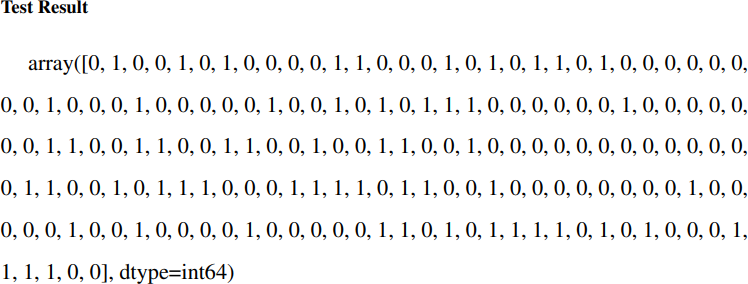
random forest and logistic regression are the best performing among all. Doing integration test gives the better idea of determine the best accuracy model among the combined ones.

## Functional Testing

Functional testing validates against the function. The main function of this testing is to test each function with different test cases. Where X value as input is given and Y values as output was verified for the respective function. Input data given as train news and test news in the respective code file. It verifies the predicted output with original output. By Functional testing, all the outputs are not correct and accurate. Based on the implemented algorithm it depends.

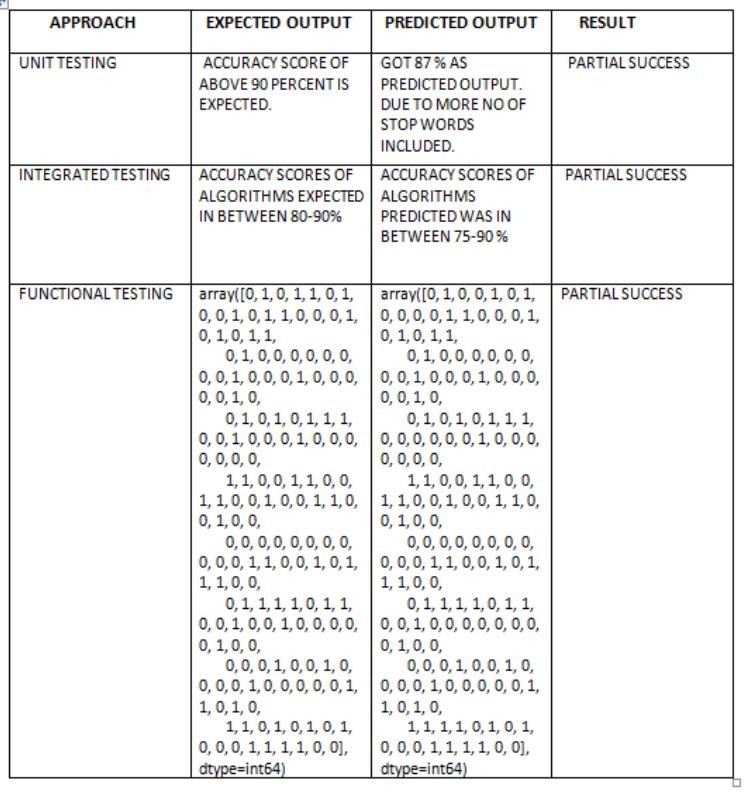
**Input**





## Test Result

**Fig 6.18 Functional Testing**



**Fig 6.19 Test Result**

## Testing Strategy

Testing is used and very important to know where our code is having any bugs/errors. Testing is implemented based on the code we implemented. Before modelling the output, it is important to maintain correct label fields assigned or not. It is also important to check whether the output matches our expectations or not. Make sure single fault in the data leads to correct output loss. Making assertions and reasons for the dataset gives us proper understanding of the code. Unit testing is implemented with pipeline concept. It helps us in verifying whether our model getting proper input or not. If not, it shows mis-fault data after few epochs training of data. Integrating testing determines the combined exact output when all the single code sets are integrated. Functional testing actually compares the predicted output with the original output to understand the results.

## CHAPTER 7 RESULTS AND DISCUSSIONS

## Efficiency of the Proposed System

The Proposed system basically implemented by the machine learning algorithms and deployed using flask. General requirements are only used which is compatible with any system or libraries installed in our PC. Based on accuracy scores final decision will be concluded. This system works efficiently for text classification type of data. The main goal of the proposed system is classifying the text and predicting the output. Different accuracy scores are generated for different algorithms. The highest accuracy score of 87.04 was generated for logistic regression and the score was really acceptable for any model with more than 10000+ records and least accuracy score of 76.08 was generated for Bernoulli naive bayes classifier and an average of 82 accuracy score was predicted for all models. Accuracy scores below 85 was not considerable further ensemble techniques need to be implemented. So, the highest will be used for final prediction of the proposed system and is used by the flask. It will take the news article as input and finally generates whether the news was true or false in web interface.

## Comparison of Existing and Proposed System

Existing system mainly focused on general old approaches to solve the problems. N grams and parts-of-speech (POS) tagging is proved insufficient in classification task and often failing to provide good context classification. Since by using deep syntax model for additional improvement it not provided the expected outcomes and not able to classify accurately. Proposed system follows the standard algorithm approaches which is basically declared as the best solving approaches and the methodology implemented provides best solution. Before implementing it. Data collected was not properly sorted. Linguistic approach used to explore, analyse and pre-process the data. The algorithms trained with different hyper parameters to achieve maximum accuracy score percentage for the taken dataset with an optimal balance between relationship exist between them and the balance between and variance and bias. Bagging, boosting is explored for evaluating to get maximum optimal results. XGBOOST classifier algorithm is trained with decision tree concept k fold cross validation is employed for all the classifier models.

## Advantages of the Proposed System

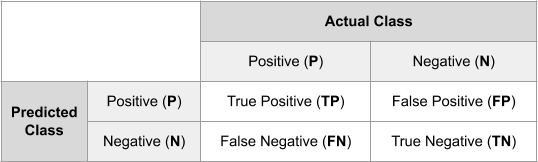
* Easy to understand and implement
* Provides Good accuracy scores
* Easily recognises patterns involved
* Handling Multi-variant data
* No human intervention required
* Detailed analysis is displayed

## Results

In this project accuracy, precision, recall, f1-Score is measured for classification of news as fake or real.

## Confusion Matrix

It is a common way of presenting true positive (TP), true negative (TN), false positive (FP) and false negative (FN) predictions. Those values are presented in the form of a matrix where the Y-axis shows the true classes while the X-axis shows the predicted classes.



**Fig 7.1 Confusion Matrix**

## Accuracy

It measures how many observations, both positive and negative, were correctly classified. Accuracy = (TP + TN) / (TP + TN + FP + FN) 7.1

## Precision

It measures how many observations predicted as positive are in fact positive.

𝑷𝒓𝒆𝒄𝒊𝒔𝒊𝒐𝒏 = 𝑻𝒓𝒖𝒆 𝑷𝒐𝒔𝒊𝒕𝒊𝒗𝒆

…………………………………………7.2

𝑻𝒓𝒖𝒆 𝑷𝒐𝒔𝒊𝒕𝒊𝒗𝒆+𝑭𝒂𝒍𝒔𝒆 𝑷𝒐𝒔𝒊𝒕𝒊𝒗𝒆

## Recall

Recall actually calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive).

𝑹𝒆𝒄𝒂𝒍𝒍 = 𝑻𝒓𝒖𝒆 𝑷𝒐𝒔𝒊𝒕𝒊𝒗𝒆 ……………………………………………..7.3

𝑻𝒓𝒖𝒆 𝑷𝒐𝒔𝒊𝒕𝒊𝒗𝒆+𝑭𝒂𝒍𝒔𝒆 𝑷𝒐𝒔𝒊𝒕𝒊𝒗𝒆

## F1 Score

The F1 score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall).

𝑭𝟏

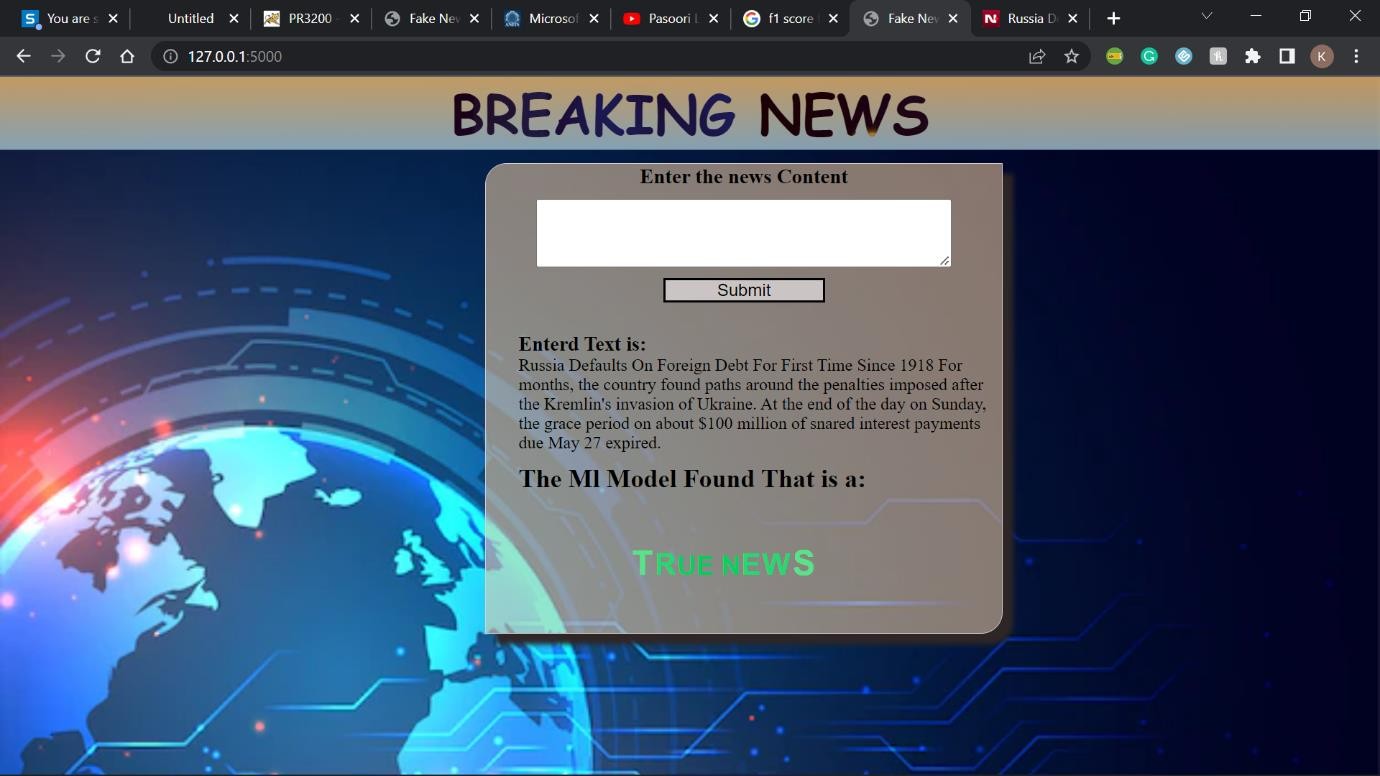
= 𝟐 ∗ 𝒑𝒓𝒆𝒄𝒊𝒔𝒊𝒐𝒏∗𝒓𝒆𝒄𝒂𝒍𝒍 …………………………………………………………….7.4

𝒑𝒓𝒆𝒄𝒊𝒔𝒊𝒐𝒏+𝒓𝒆𝒄𝒂𝒍𝒍

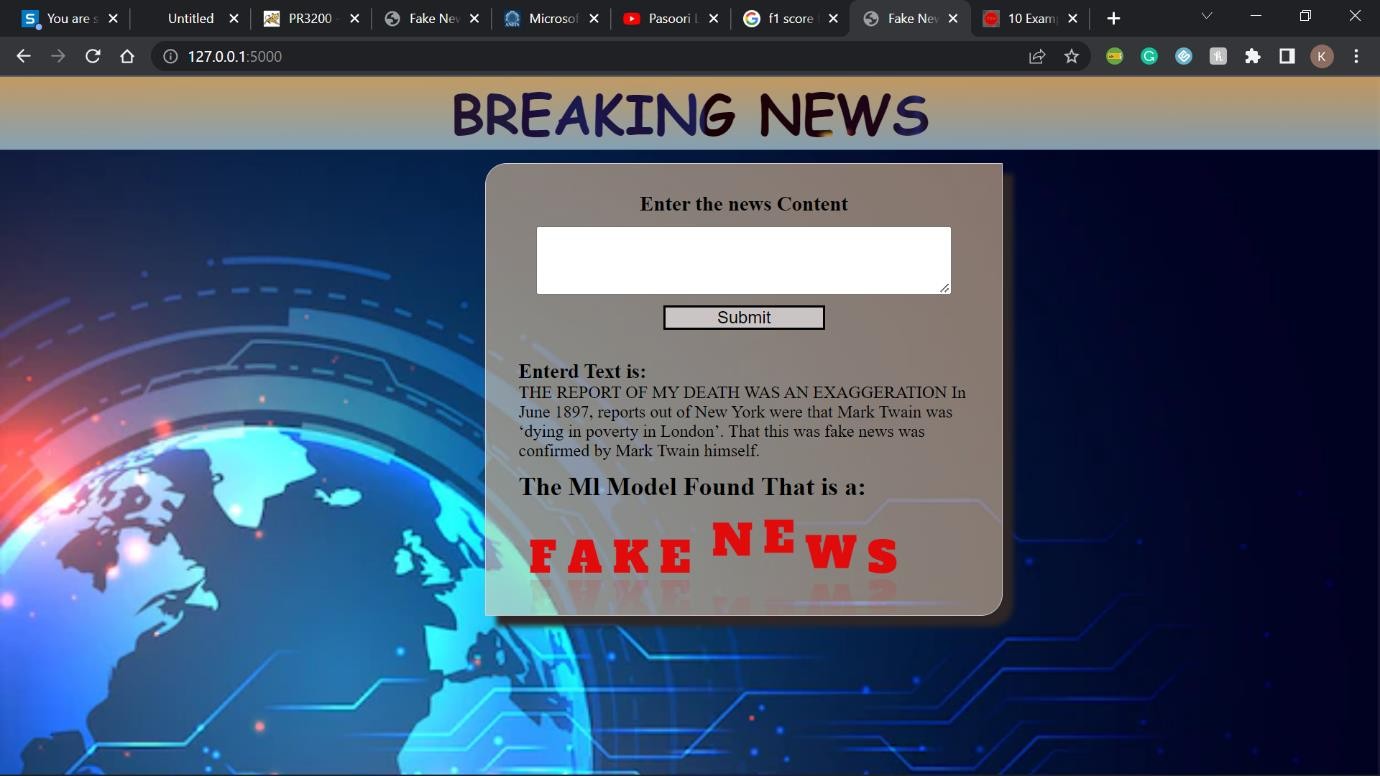
## Model Deployment

This is what you see when you go to the web interface. You are supposed to copy the news and paste it into the input box.

When you paste the news on the input box and click ‘Submit’ the model will give you the result. If the news seems authentic, the output will be ‘True News’. Otherwise, it will show ‘Fake News’. That’s how you can detect fake or real news via the interface.



**Fig 7.2 Output page for True news**



**Fig 7.3 Output page for fake news**

# CHAPTER 8

**CONCLUSION AND FUTURE ENHANCEMENTS**

## Conclusion

The popularity of social media increased so more and more news consuming from the social media rather than the traditional news channels. However, social media is faster to spread false news among the people and to create negative impact on broader society as well as on individual user. In this project fake news has been explored by reviewing existing system in two phases: characterization and detection. In the first phase introduction basic concepts of fake news and principles of it on social media and traditional media. Second phase work is based on reviewing and analysing on existing system using data mining perspective and extracting feature extraction models and model construction. With the data collected from WhatsApp, Facebook, twitter etc the best models improve every day of a week. Also, here discussed about the various datasets used and how to implement algorithms on various applications. This project gives the clear solution of how to identify and detect the fake news. The limitations here we got is that data is erratic means every type of algorithm is not fully perfect. It has anomalies and can make mistakes.

## Future Enhancements

However, the use of all the available algorithms not benefitted in producing more accuracy scores. Many efforts have been taken in increasing prediction scores but the result has yet to be seen. Increase in data limit gives more stop words and tendency to remove those will become very difficult for the classifiers. Deep neural networks concepts help in creating model with lot more promising. For future enhancements, techniques like neural networks, word2vec, topic modelling, POS tagging can be used. These techniques will give lot more depth analysis and good feature extraction and well-tuned classification model. Deep learning classification provides good accuracy results.

# REFERENCES

* + 1. Choudhary, M., Jha, S., Saxena, D., & Singh, A. K. (2021, May). A Review of Fake News Detection Methods using Machine Learning. In *2021 2nd International Conference for Emerging Technology (INCET)* (pp. 1-5). IEEE.
    2. Sharma, U., Saran, S., & Patil, S. M. (2020). Fake news detection using machine learning algorithms. *International Journal of Creative Research Thoughts (IJCRT)*, *8*(6), 509-518.
    3. Nada, F., Khan, B. F., Maryam, A., Zuha, N., & Ahmed, Z. (2019). Fake news detection using logistic regression. *International Research Journal of Engineering and Technology (IRJET).* [*https://w*](http://www/)*ww. irjet. net/archives*, *6*.
    4. Singh, G., Kumar, B., Gaur, L., & Tyagi, A. (2019, April). Comparison between multinomial and Bernoulli naïve Bayes for text classification. In *2019 International Conference on Automation, Computational and Technology Management (ICACTM)* (pp. 593-596). IEEE.
    5. Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, *19*(1), 22- 36.
    6. Granik, M., & Mesyura, V. (2017, May). Fake news detection using naive Bayes classifier. In *2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON)* (pp. 900-903). IEEE.
    7. Shetty, B. (2018). Natural Language Processing (NLP) for machine learning. *Retrieved November*, *24*, 2018.
    8. What is a Confusion Matrix in Machine Learning by Jason Brownlee on November 18, 2016 in Code Algorithms From Scratch.
    9. Mahabub, A. (2020). A robust technique of fake news detection using Ensemble Voting Classifier and comparison with other classifiers. *SN Applied Sciences*, *2*(4), 1-9.
    10. Shelke, S., & Attar, V. (2019). Source detection of rumor in social network–a review. *Online Social Networks and Media*, *9*, 30-42.
    11. Velay, M., & Daniel, F. (2018). Using NLP on news headlines to predict index trends. *arXiv preprint arXiv:1806.09533*.
    12. Kogan, S., Moskowitz, T. J., & Niessner, M. (2019). Fake news: Evidence from financial markets. *Available at SSRN*, *3237763*.
    13. Rubin, V. L., Conroy, N., Chen, Y., & Cornwell, S. (2016, June). Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the second workshop on computational approaches to deception detection* (pp. 7-17).