FINDING INCONSISTENCY OF SECURITY INFORMATION FROM UNSTRUCTURED TEXT

A PROJECT REPORT

Submitted by

SANGGAMI R [REGISTER NO:211417104238]

PUNITHA G [REGISTER NO:211417104205]

SARANYA R [REGISTER NO:211417104244]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

AUGUST 2021

BONAFIDE CERTIFICATE

Certified that this project report **"FINDING INCONSISTENCY OF SECURITY INFORMATION FROM UNSTRUCTURED TEXT"** is the bonafide work of **"SANGGAMI R [REGISTER NO: 211417104238], PUNITHA G [REGISTER NO: 211417104205] and SARANYA R [REGISTER NO: 211417104244]"** who carried out the project work under mysupervision.

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,

NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mrs. SANGEETHA K,M.E.,
Assistant Professor

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on05.08.2021

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to express our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR, M.E., Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my Project Guide,**Mrs. SANGEETHA K,M.E.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

SANGGAMI R
PUNITHA G
SARANYA R

ABSTRACT

Open-source intelligence textual data has become a big topic in a variety of fields, including industry, law enforcement, military, and cyber security. Millions of people use social networking sites all over the world. The user's experiences with social media platforms like Twitter and Face book have a huge effect on their everyday lives, with some detrimental effects. The identification of scams and phony social media users has become a common research area in the field of modern online social networks (OSNs). We show that Supervised Machine learning can overcome all the bottlenecks. We here find the conflict, disagreement or mismatch of the secured information from the unstructured data. Text analyses and classification is made to detect the Cyber bullying activities, Fake News Identification and Spamming.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

S. NO.	ABBREVATION	EXPANSION
1	ML	Machine Learning
2	CS	Computational Stylometry
3	CNN	Convolutional Neural Network
4	LSTM	Long Short-Term Neural Network
5	GCN	Graph Convolutional Network
6	SIF	Smooth Inverse Frequency
7	NLP	Natural Language Processing
8	SVM	Support Vector Machine
9	IT	Information Technology
10	RAM	Random Access Memory
11	GB	Giga Byte
12	GHz	GigaHertz
13	SGD	Stochastic Gradient Descent

14	DBSCAN	Density-Based Spatial Clustering of Applications with Noise
15	OPTICS	Ordering Points to Identify Clustering Structure
16	CLARANS	Clustering Large Applications based upon randomized Search
17	STING	Statistical Information Grid
18	CLIQUE	CLustering In Quest
19	MDP	Markov Decision Process
20	Numpy	NUMeric Python
21	Pytest	Python TESTing
22	Sympy	SYMbolic Python
23	Scipy	SCIentific Python
24	GUI	Graphical User Interface
25	TF	Term Frequency
26	IDF	Inverse Document Frequency
27	TF-IDF	Term Frequency - Inverse Document Frequency
28	BOW	Bag Of Words
29	KNN	K - Nearest Neighbor
30	TP	True Positive
31	TN	True Negative
32	FP	False Positive
33	FN	False Negative
34	PR	Precision- Recall
35	ROC	Receiver Operator Characteristics curve
36	OSNs	Online Social Networks

CHAPTER - 1

CHAPTER 1

INTRODUCTION

Internet usage as increased tremendously. It is been quite unpretentious to get any sort of information from any kind of source around the world. The demand for resources is high from online networking sites since networking sites have become an online source to collect real-time data about any users. Users can share anything, and get information about any sectors and fields. Besides this, it also provides comments for the way of information discrimination which is never been witnessed earlier. Many people who do not have much idea regarding online networking can be tricked easily by fraudsters. With the growth of Online networking sites, there needs to be analysis and study of user behaviour.

1.1OVERVIEW

As the usage of the internet increases highly, it has both positive and negative side impacts. There is more fraudsters activity taking place. This activity leads to many threats, attacks. As the data available and shared on the internet in the form of text, video, audio, and so on. We here perform text classification and analyse the text. Text classification is necessary due to the large number of text documents that we need to deal with daily. Here, we detect cyberbullying activities, fake news, and spamming detection. Since much online harassment and abusive activities occur, we look into cyberbullying detection. Spreading of fake news has become common which leads to spreads many misleading pieces of information and influence society by manipulating with help of it, so fake news detection is carried out. Unwanted and unsolicited information has been digitally communicated which is been irrelevant and unreliable so detection of spamming is done here.

1.2PROBLEM DEFINITION

The objective is to overcome the inconsistency of the information in the form of text that is available on the internet. The inconsistency in the text may lead to threats, attacks, and negative effects by the fraudsters to the users. To overcome the threat-free and reduce negative effects text classification and analysis are made and detection is performed. This is done by a collection of datasets in a structured format, then pre-processing the text and bring into the corpus (collection of data), and then vectorize the corpus to convert the text into vector form. By doing this the feature extraction is made and the with help of supervised machine learning model classification and analysis is made. Then the accurate prediction is determined. By doing this we can predict which model will give a higher accuracy rate. The model with high accuracy can help to detect fake news identification, spamming, and cyberbullying activities effectively.

CHAPTER - 2

CHAPTER 2

LITERATURE SURVEY

1. **TITLE -** Gender and Age Detection in Cyber bullying Texts Using Computational Stylometry and Machine Learning

AUTHOR - Antonio Pascucci; Vincenzo Masucci; Johanna Monti

YEAR - 2019

DESCRIPTION -The purpose of this paper is to demonstrate the value ofComputational Stylometry (CS) and Machine Learning (ML) in detecting author gender and age in cyber bullying texts. We created a cyber bullying detection platform and show the results of gender and age detection in cyber bullying texts we collected in terms of Precision, Recall, and F-Measure.

MERITS - We can categorize and analyse cyber bullying text by grouping it into taxonomy (linguistic rules) to detect cyber bullies' gender and age.

DEMERITS –

- Focusing on categories based solely on written text.
- Developing a smaller number of stylistic features.

2. **TITLE** - Spam detection in social media using convolutional and long short-term memory neural network

AUTHOR - Gauri Jain; Manisha Sharma ; Basant Agarwal

YEAR –2019

DESCRIPTION - People are virtually connected as the use of the Internet grows, using social media platforms such as text messages, Facebook, Twitter, and other similar platforms. This has resulted in an increase in the spread of spam, or unsolicited messages, which are used for marketing, data collection, or simply to offend people. As a result, having a strong spam detection architecture that can prevent these types of messages is critical. Due to the short text and high variability in the

language used in social media, spam detection on noisy platforms like Twitter remains a challenge. We propose a novel deep learning architecture based on Convolutional Neural Networks (CNN) and Long Short-Term Neural Networks (LSTN) in this paper (LSTM). With the help of knowledge bases such as WordNet and ConceptNet, the model is supported by introducing semantic information in the representation of words. The use of these knowledge bases improves performance by providing a better semantic vector representation of testing words that had a random value before because they were not seen in the training. Experimental results on two benchmark datasets show that the proposed approach is effective in terms of accuracy and F1-score.

MERITS -The proposed approach highly helpful in detecting the spam in short noise text – a challenging one due to sparseness and irregularities of language.

DEMERITS – Overfit especially to the low volume of data

3. **TITLE -** An Unsupervised Approach of Truth Discovery from Multi-sourced Text Data.

   AUTHOR - Chen Chang; Jianjun Cao; Qibin Zheng

   YEAR - 2018

   DESCRIPTION -Truth discovery methods aim to identify which piece of information is trustworthy from multi-sourced data. Most existing truth discovery methods, however, are designed for structured data and fail to meet the strong need to extract trustworthy information from raw text data. More specifically, existing methods ignore the semantic information of text answers, i.e., answers may contain multiple factors, the word usages may be diverse, and the answers may be partially correct. In

addition, ubiquitous long-tail phenomenon exists in the tasks, i.e., most users provide only a few answers and only a few users provide plenty of answers, which causes the user reliability estimation for small users to be unreasonable. To tackle these challenges, we propose a Graph Convolutional Network (GCN) based truth discovery model to automatically discover trustworthy information from text data. Firstly, Smooth Inverse Frequency (SIF) is utilized to learn real-valued vector representations for answers. Then, we construct undirected graph with these vectors to capture the structural information of answers. Finally, the GCN is utilized to store and update the reliability of these answers, and sums up all the feature vectors of all neighbouring answers to improve the accuracy and efficiency of truth discovery. Different from traditional methods, we use vectors to store the reliability of answers which have higher representation capability compared with real numbers, and network is utilized to capture complex relationships among answers rather than simplified functions. The experiment results on real datasets show that though text data structures are complex, our model can still find reliable answers compared with retrieval-based and state-of-the-art truth discovery methods.

MERITS –

- Easily identifies which piece of information is trustworthy from structured data.
- It achieves the best performance by capturing the semantic meaning of the text data.

DEMERITS –

- It fails to meet the strong need to extract trustworthy information from raw text data.
- Less effective for some complex situations such as thenumber of answers to each question is not enough.

4. **TITLE -** An integrated approach for malicious tweets detection using NLP

AUTHOR -Sagar Gharge; Manik Chavan

YEAR -2017

DESCRIPTION -Detection of malicious user accounts has been the focus of many previous studies. Detecting spam or spammers on Twitter is a relatively new area of social network research. However, we present a method based on two new aspects: the detection of spam-tweets without knowing the user's previous background, and the other based on language analysis for detecting spam on Twitter in topics that are currently trending. Topics of discussion that are popular at the time are known as trending topics. Spammers benefit from the growing micro blogging phenomenon. Our research uses language tools to detect spam tweets. We began by gathering tweets related to a variety of popular topics and categorizing them as malicious or non-malicious. We extracted a number of features based on language models using language as a tool after a labeling process. We also assess performance and categorize tweets as spam or not. As a result, our system can be used to detect spam on Twitter, with a focus on tweet analysis rather than user accounts.

MERITS - We detect spam accounts using the SVM algorithm, which yields a standard result.

DEMERITS - It can only be used to detect spam on Twitter, with a focus on tweet detection.

5. **TITLE -** Unsupervised cyber bullying detection in social networks

AUTHOR - Michele Di Capua; Emanuel Di Nardo; Alfredo Petrosino

YEAR - 2016

DESCRIPTION - Modern young people (also known as "digital natives") have grown up in a world dominated by new technologies, in which communication is pushed to near-real-time levels and there are no boundaries to forming relationships with other people or communities. However, due to the rapid pace of evolution, young people are unable to distinguish between consciously acceptable and potentially harmful behaviours, and a new phenomenon known as cyber bullying is gaining traction, attracting the attention of educators and the media. “Willful and repeated harm inflicted through the use of electronic devices” is what cyber bullying is defined as. Using techniques derived from NLP (Natural Language Processing) and machine learning, we propose a possible solution for automatic detection of bully traces across a social network in this paper. We will create a model based on Growing Hierarchical SOMs that can efficiently cluster documents containing bully traces and is based on semantic and syntactic features of textual sentences. We fine-tuned our model to work with Twitter, but we also put it to the test with other social media platforms like YouTube and Form spring. Finally, we present our findings, demonstrating that the proposed unsupervised approach can be used effectively in some scenarios with good results.

MERITS – Use of an unsupervised approach to analyse cyber bullying with several handcrafted features that were used to catch the cyber bullies' semantic and syntactic communication behaviour.

DEMERITS - Less accuracy and performance results.

CHAPTER - 3

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Textual data mining of open-source intelligence onthe Web has become an increasingly important topic acrossa wide range of domains such as business, law enforcement,military, and cybersecurity. Text mining efforts utilize naturallanguage processing to transform unstructured web content intostructured forms that can drive various machine learning applications and data indexing services. For example, applicationsfor text mining in cybersecurity have produced a range ofthreat intelligence services that serve the IT industry. However,a less studied problem is that of automating the identificationof semantic inconsistencies among various text input sources.In this paper, we introduce GapFinder, a new inconsistencychecking system for identifying semantic inconsistencies withinthe cybersecurity domain. Specifically, we examine the problemof identifying technical inconsistencies that arise in the functionaldescriptions of open-source malware threat reporting information. Our evaluation, using tens of thousands of relations derivedfrom web-based malware threat reports, demonstrates the abilityofGapFinder to identify the presence of inconsistencies.

3.2 PROPOSED SYSTEM

The spread of fake news on the Internet and cyber bullying activities on social media is a cause of great concern for all members of society, including the government, policymakers, organizations, businesses, and citizens. Fake news spread through social media in the country has become a serious problem, with the potential of it resulting in mob violence. Bullying can impact a person on physical, mental, and emotional levels which leads to depression. Spam emails thus run the risk of their computer being infected by a malicious program.

Adeluge of spam will clog mail servers, making all email slow and burdening the ISP. Since spam email, fake news detection and cyber bullying activities comes under text analysis we have combined all these domains and developed a system that is able to detect such texts would surely be of great use. In our proposed system we perform text classification on the dataset to identify fake/true news, spam/ham in the mail, bullying and non-bullying activity by using a supervised machine learning algorithm. The real-time dataset is collected before applying it directly to the algorithm, we clean and prepare the data, by using some pre-processing techniques like stemming, lemmatization, and feature extraction is made to derive values (features in vector format) intended to be informative and non-redundant. Applying features to different supervised machine learning models accuracy is predicted for each model and the one which has high accuracy is effective to detect the problem,

3.3TECHBOLOGY STACK

HARDWARE REQUIREMENTS

Processor : Intel Pentium Dual Core 2.00GHz

Hard disk : 500 GB

RAM : 8 GB (minimum)

SOFTWARE REQUIREMENTS

Python 3.6.4 Version

3.4 SOFTWARE SPECIFICATION

Machine learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer

vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

The types of machine learning algorithms are mainly divided into four categories:

- **Supervised learning,**
- **Un-supervised learning,**
- **Semi-supervised learning,**
- **Reinforcement learning**.

3.4.1 Supervised learning

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Classification

As the name suggests, Classification is the task of "classifying things" into sub- categories. But, by a machine. If that doesn't sound like much, imagine your computer being able to differentiate between you and a stranger. Between a potato and a tomato. Between an A grade and a F. In Machine Learning and Statistics, Classification is the problem of identifying to which of

a set of categories (sub populations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

Types of Classification

Classification is of two types:

- Binary Classification
- Multiclass Classification

Binary Classification

When we have to categorize given data into 2 distinct classes. Example – On the basis of given health conditions of a person, we have to determine whether the person has a certain disease or not.

Multiclass Classification

The number of classes is more than 2. For Example,On the basis of data about different species of flowers, we have to determine which specie does our observation belong to Binary and Multiclass Classification. Here x1 and x2 are our variables upon which the class is predicted.Suppose we have to predict whether a given patient has a certain disease or not, on the basis of 3 variables, called features. Which means there are two possible outcomes:

1.The patient has the said disease. Basically, a result labelled “Yes” or “True”.

2.The patient is disease free. A result labelled “No” or “False”.

This is a binary classification problem.We have a set of observations called training data set, which comprises of sample data with actual classification results.

1. X: pre-classified data, in the form of a N*M matrix. N is the no. of observations and M is the number of features

2. y: An N-d vector corresponding to predicted classes for each of the N observations.

3. Feature Extraction: Extracting valuable information from input X using a series of transforms.

4. ML Model: The “Classifier” we’ll train.

5.y’: Labels predicted by the Classifier.

6.Quality Metric: Metric used for measuring the performance of the model.

7.ML Algorithm: The algorithm that is used to update weights w’, which update the model and “learns” iteratively.

Types of Classifiers (Algorithms)

There are various types of classifiers. Some of them are:

- Linear Classifiers: Logistic Regression
- Tree Based Classifiers: Decision Tree Classifier
- Support Vector Machines
- Artificial Neural Networks
- Bayesian Regression
- Gaussian Naive Bayes Classifiers
- Stochastic Gradient Descent (SGD) Classifier
- Ensemble Methods: Random Forests, AdaBoost, Bagging Classifier, Voting Classifier, Extra Trees Classifier

Practical Applications of Classification

- Google’s self-driving car uses deep learning enabled classification techniques which enables it to detect and classify obstacles.

- Spam E-mail filtering is one of the most widespread and well recognized uses of Classification techniques.
- Detecting Health Problems, Facial Recognition, Speech Recognition, Object Detection, Sentiment Analysis all use Classification at their core.

Regression

A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. Many different models can be used, the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

3.4.2 Un-supervised learning

Un-supervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Clustering

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples. Clustering is

same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.For example, the data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture

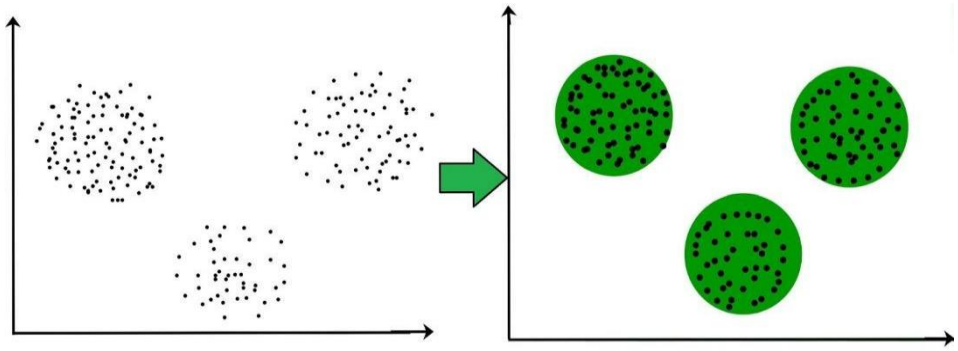


Fig No. 3.1 Distinguish cluster to identify three kind of cluster

These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Various distance methods and techniques are used for calculation of the outliers.

Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties ("natural" data types), in finding useful and suitable groupings ("useful" data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

Clustering Methods:

1. **Density-Based Methods:** These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters. Example DBSCAN - (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure) etc.
2. **Hierarchical Based Methods:** The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two categories
   - •Agglomerative (bottom-up approach)
   - •Divisive (top-down approach).
3. **Partitioning Methods:** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example K-means, CLARANS (Clustering Large Applications based upon randomized Search) etc.
4. **Grid-based Methods:** In this method the data space are formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example STING (Statistical Information Grid), wave cluster, CLIQUE (CLusteringIn Quest) etc.

Clustering Algorithms:

- K-Means Clustering.
- Mean-Shift Clustering for a single sliding window.

- The entire process of Mean-Shift Clustering.
- DBSCAN Smiley Face Clustering.
- EM Clustering using GMMs.
- Agglomerative Hierarchical Clustering.

3.4.3 Semi-supervised learning

Semi-supervised learning falls between unsupervised learning (without any labelled training data) and supervised learning (with completely labelled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabelled data, when used in conjunction with a small amount of labelled data, can produce a considerable improvement in learning accuracy.

3.4.4 Reinforcement learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, information theory, simulation-based optimization, multi-agent system, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcement learning algorithms usedynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

ANACONDA

Anaconda is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific

computing), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. Anaconda Distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

Python is a high-level programming language devised by Guido van Rossum & first released in 1991. It's the most popular coding language used by software developers to build, control, manage and for testing. It is also an interpreter which executes Python programs. The python interpreter is called python.exe on Windows.

Python Packages

Packages or additional libraries help in scientific computing and computational modelling. In Python, the packages are not the part of the Python standard library. Few major packages are –

numpy (NUMeric Python): matrices and linear algebra

scipy (SCIentific Python): many numerical routines

matplotlib: (PLOTtingLIBrary) creating plots of data

sympy (SYMbolic Python): symbolic computation

pytest (Python TESTing): a code testing framework

Together with a list of Python packages, tools like editors, Python distributions include the Python interpreter. Anaconda is one of several Python distributions. Anaconda is a new distribution of the Python and R data science package. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages.

This work environment, Anaconda is used for scientific computing, data science, statistical analysis, and machine learning. The latest version of Anaconda 5.0.1 is released in October 2017.The released version 5.0.1 addresses some minor bugs and adds useful features, such as updated R anguage support. All of these features weren't available in the original 5.0.0 release.

This package manager is also an environment manager, a Python distribution, and a collection of open-source packages and contains more than 1000 R and Python Data Science Packages.

IPYTHON NOTEBOOKS

IPython is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language, that offers introspection, rich media, shell syntax, tab completion, and history. IPython provides the following features:

- A browser-based notebook interface with support for code, text, mathematical expressions, inline plots and other media.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into one's own projects.
- Tools for parallel computing.

IPython is based on an architecture that provides parallel and distributed computing. IPython enables parallel applications to be developed, executed, debugged and monitored interactively. Hence, the I (Interactive) in IPython.This architecture abstracts out parallelism, which enables IPython to support many different styles of parallelismincluding:

With the release of IPython 4.0, the parallel computing capabilities have been made optional and released under the ipyparallel python package.IPython frequently draw from SciPy stack libraries like NumPy and SciPy, often installed alongside from one of many Scientific Python distributions.

Other features:

IPython also allows non-blocking interaction with Tkinter, PyGTK, PyQt/PySide and wxPython (the standard Python shell only allows interaction with Tkinter). IPython can interactively manage parallel computing clusters

using asynchronous status call-backs and/or MPI. IPython can also be used as a system shell replacement. Its default behaviour is largely similar to Unix shells, but it allows customization and the flexibility of executing code in a live Python environment. Using IPython as a shell replacement is less common and it is now recommended to use Xonsh which provide most of the IPython feature with better shell integrations.

JUPYTER NOTEBOOK

Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document.

Jupyter Notebook provides you with an easy-to-use, interactive data science environment across many programming languages that doesn't only work as an IDE, but also as a presentation or education tool.

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

CHAPTER - 4

CHAPTER 4

SYSTEM DESIGN

4.1 UML DIAGRAM

USE CASE

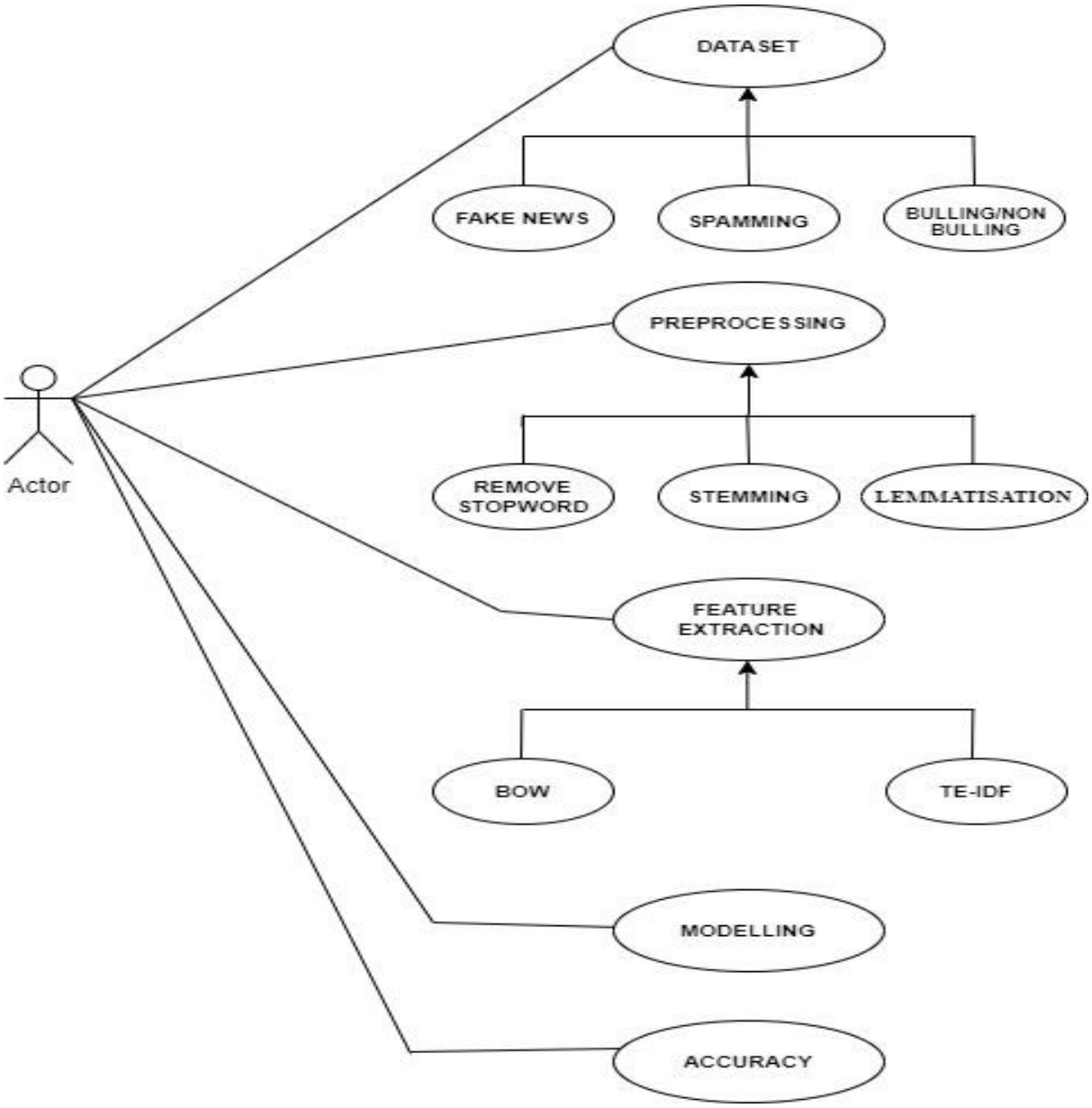


Fig. No. 4.1 Use case diagram

ACTIVITY DIAGRAM

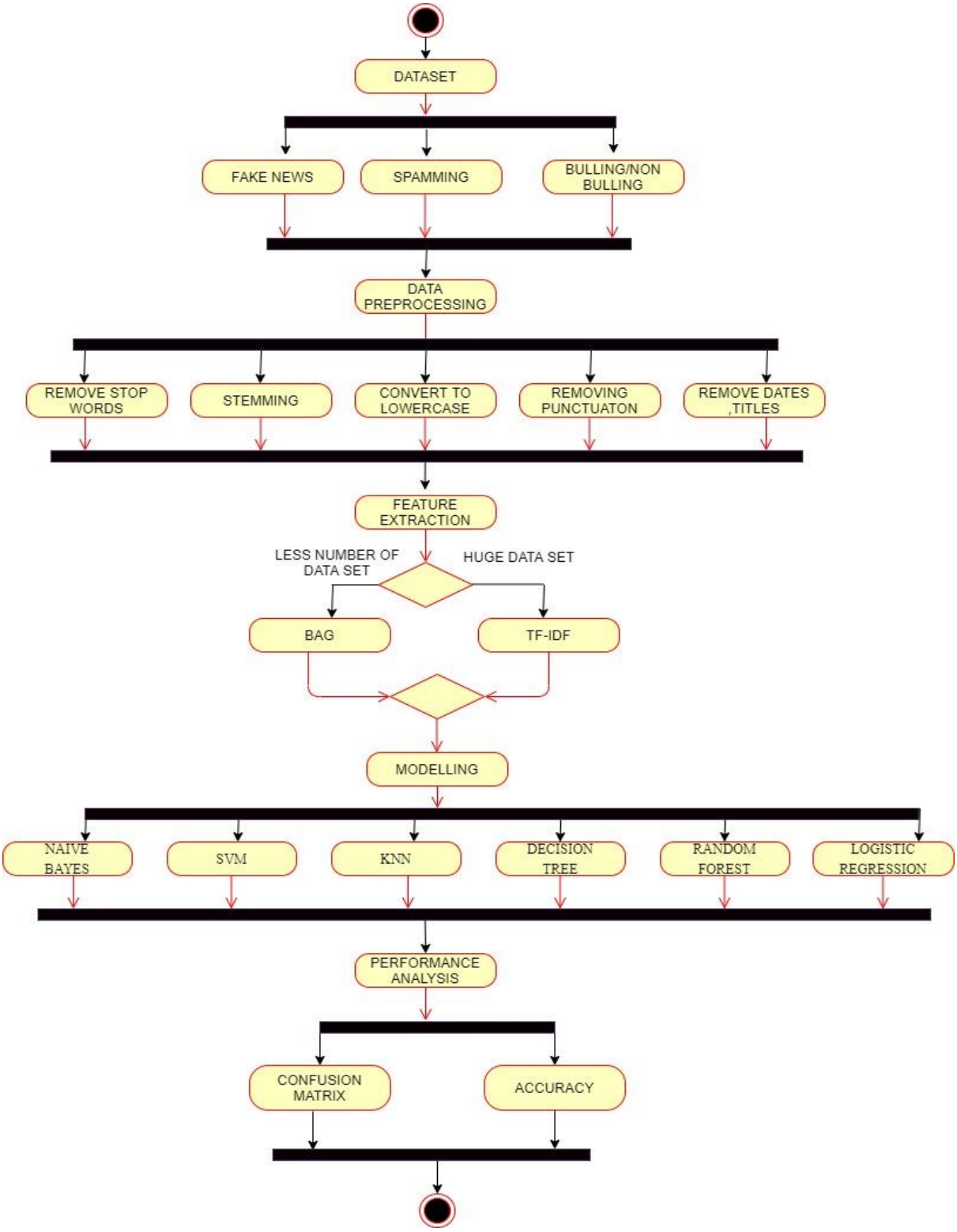


Fig. No. 4.2 Activity Diagram

SEQUENCE DIAGRAM

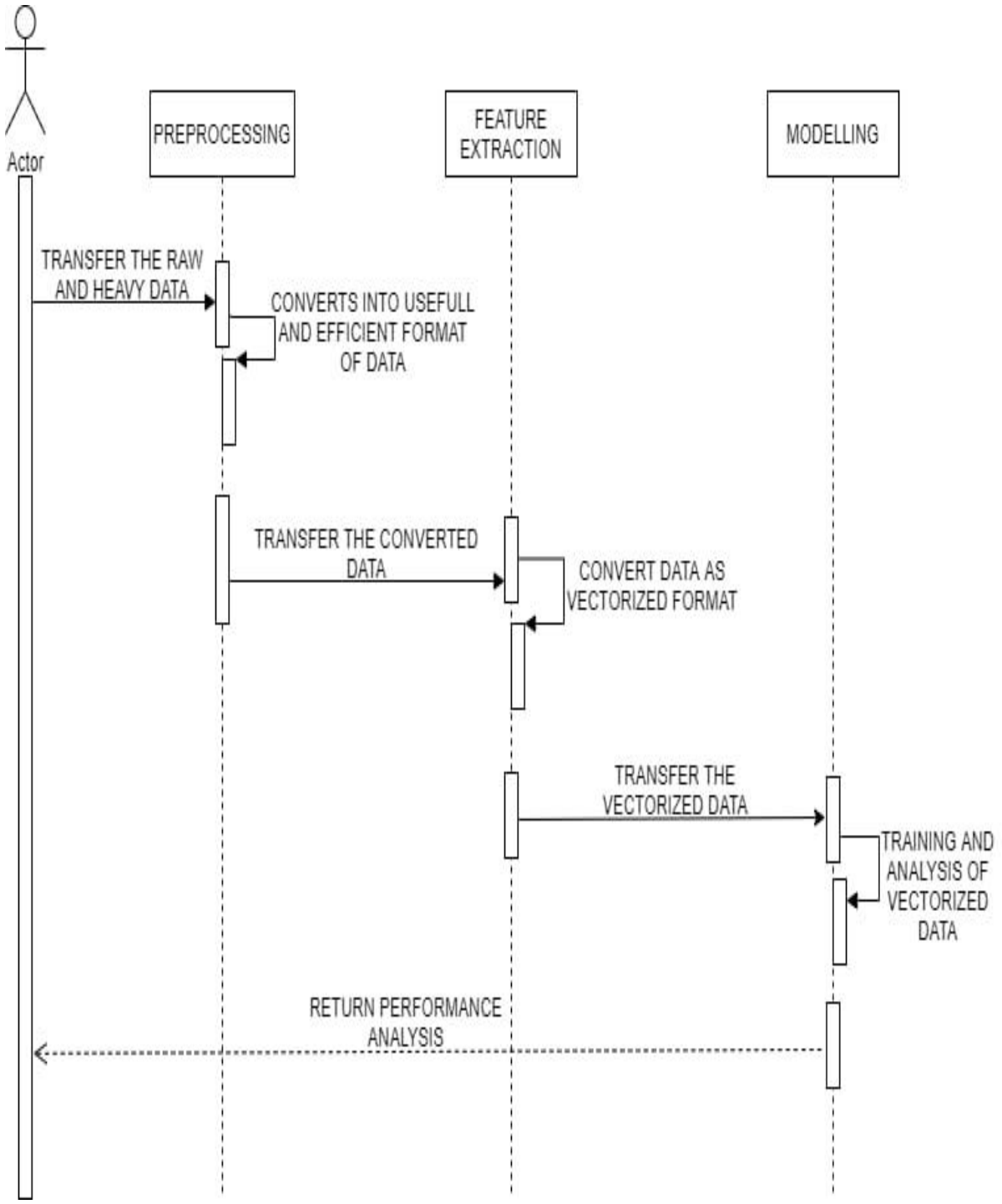


Fig. No. 4.3 Sequence Diagram

CHAPTER - 5

CHAPTER 5

SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. This system provides majorly four modules, which are mentioned below. The system provides user friendly GUI and data flows to all modules in dynamic way.

5.1 ARCHITECTURE OVERVIEW

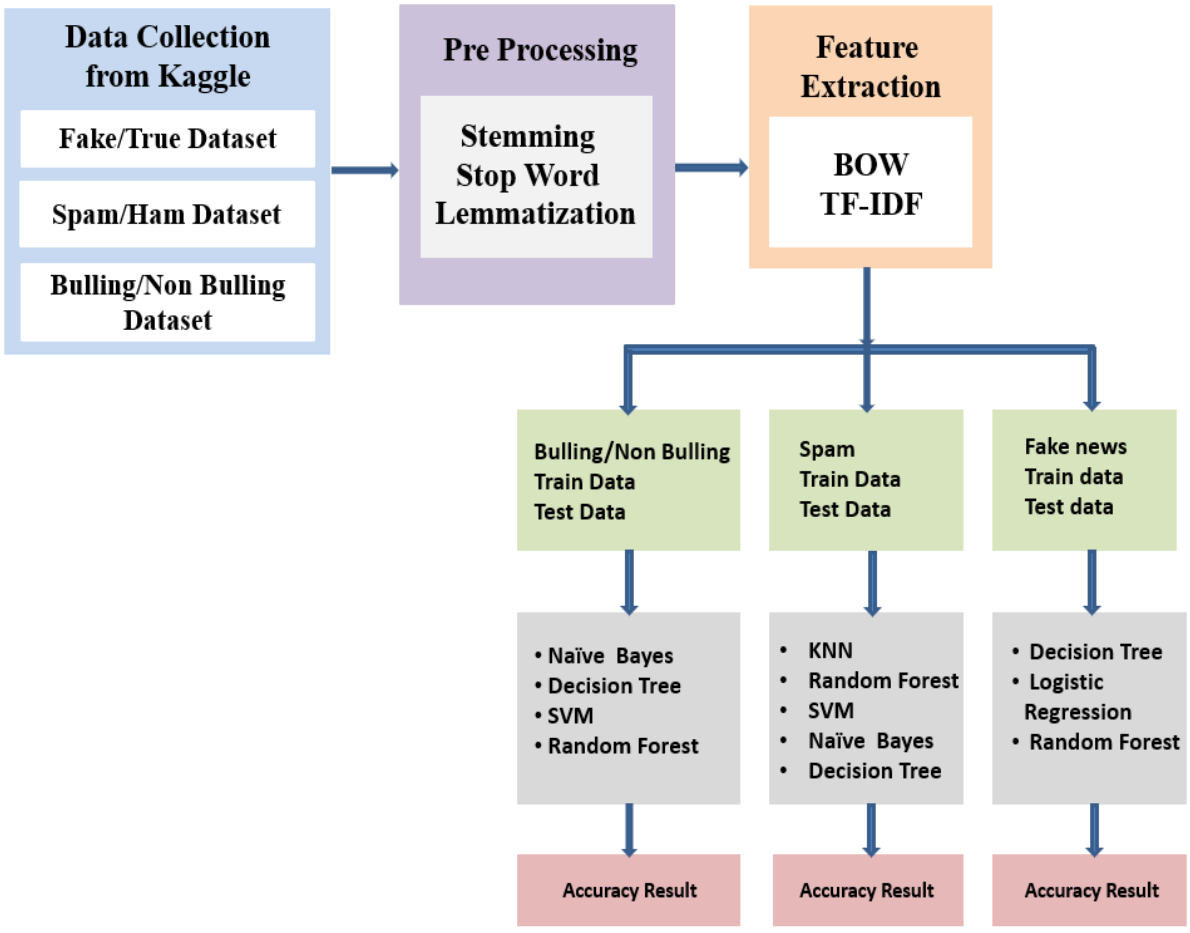


Fig. No. 5.1 System Architecture

Real-time Unstructured data is made into structured format are collected from Kaggle. We are collecting datasets specifically for Bullying, Fake News, and Spamming. Then the datasets collected will be subjected to pre-processing like Stemming, Stop Word and tokenization if needed lemmatization. To analyse a pre-processed data, it needs to be converted into features i.e., the text must be converted into vectors of numbers. A technique for extracting features from text is by BOW and TF - IDF. A bag of words is a method to represent text into numerical features. Bag of Features can be easily created by using the count Vectorizer function. This process is called feature extraction (or Vectorization). It is used to convert a collection of text documents to a vector of token counts. TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. Then we will be building predictive models for Bullying, Fake News, and Spamming. For bullying, we will be using Decision Tree, Linear Regression, and Random Forest. For Spamming, we will be using Support Vector Machine, Naïve Bayes, Decision Tree, Random Forest, and K - Nearest Neighbor. Then for fake news, we will be using Random Forest, Linear Regression, and Decision Tree. At last, based on the outcomes we will predict the probability of occurrence of an event by evaluating Metrics Accuracy. We conclude that the model with good accuracy is chosen as the best one.

5.2 MODULE DESIGN SPECIFICATION

5.2.1 MODULE

- Data Collection.
- Data Pre-processing.
- Data Exploration
- Feature Extraction
- Modelling

5.2.2 MODULE DESCRIPTION

5.2.2.1 DATA COLLECTION

- Real time unstructured data which is been modified to structured format is collected from Kaggle.
- Peculiar datasets is collected for each (i.e.,) bullying and non-bullying, Fake news, Spamming.
- Collection of data is most important tasks of any machine learning projects. Because the input we feed to the algorithms is data.
- So, the algorithms efficiency and accuracy depend upon the correctness and quality of data collected

	File Format	**Data Count**	**Source**
Cyber Bullying	JSON	20001	Kaggle
Spamming	CSV	Fake = 23503	Kaggle
		True = 21418	
Fake News	CSV	5573	Kaggle

Table No.5.1 Data Samples

5.2.2.2DATA PRE-PROCESSING

Data Pre-processing, we transform the raw data in a useful and efficient format by removing noise and inconsistent data like Punctuation, date, title, Stop Words, Stemming, Converting the words to Lowercase and if needed lemmatization. So, it creates the reliable consistent data that improves

theefficiency of the training data for analysis and also enables accurate decision making.

	title	text	subject	date	target
0	Croatia wants to adopt euro within 7-8 years: ...	ZAGREB (Reuters) - Croatia aims to become a eu...	worldnews	October 30, 2017	true
1	House narrowly passes measure paving way for T...	WASHINGTON (Reuters) - The U.S. House of Repre...	politicsNews	October 26, 2017	true
2	Oklahoma rejects Russian request to monitor el...	(Reuters) - Oklahoma voting officials have den...	politicsNews	October 21, 2016	true

Fig. No. 5.2 Preprocessing data

1. Removing Date (attribute not used for analysis)

	title	text	subject	target
0	Croatia wants to adopt euro within 7-8 years: ...	ZAGREB (Reuters) - Croatia aims to become a eu...	worldnews	true
1	House narrowly passes measure paving way for T...	WASHINGTON (Reuters) - The U.S. House of Repre...	politicsNews	true
2	Oklahoma rejects Russian request to monitor el...	(Reuters) - Oklahoma voting officials have den...	politicsNews	true

Fig. No. 5.3 Removing Date (unwanted attribute)

2. Removing the title (we will only use the text)

	text	subject	target
0	ZAGREB (Reuters) - Croatia aims to become a eu...	worldnews	true
1	WASHINGTON (Reuters) - The U.S. House of Repre...	politicsNews	true
2	(Reuters) - Oklahoma voting officials have den...	politicsNews	true

Fig. No. 5.4 Removing Title

3. Convert to lowercase

	text	subject	target
0	zagreb (reuters) - croatia aims to become a eu...	worldnews	true
1	washington (reuters) - the u.s. house of repre...	politicsNews	true
2	(reuters) - oklahoma voting officials have den...	politicsNews	true

Fig. No. 5.5 Pre-processing convert to lowercase

4. **Remove punctuation**

	text	subject	target
0	zagreb reuters croatia aims to become a euro ...	worldnews	true
1	washington reuters the us house of representa...	politicsNews	true
2	reuters oklahoma voting officials have denied...	politicsNews	true

Fig. No. 5.6 Pre-processing – removing punctuation

5. **Removing stopwords**

	text	subject	target
0	zagreb reuters croatia aims become euro zone m...	worldnews	true
1	washington reuters us house representatives he...	politicsNews	true
2	reuters oklahoma voting officials denied reque...	politicsNews	true
3	world reeling today britain made shocking deci...	News	fake

Fig. No. 5.7 Pre-processing – removing stopwords

6. **Stemming**

Stemming just removes or stems the last few characters of a word, often leading to incorrect meanings and spelling.

e.g., adjustable - > adjust

formality - > formal

7. **Lemmatisation**

Lemmatization considers the context and converts the word to its meaningful base form.

e.g., playing, plays, played - > play

am, are, is -> be

5.2.2.3 DATA EXPLORATION

We use visual exploration to understand what is in a dataset and the character rustics of the data. Here the characteristics are size, amount of data, completeness of data, correctness of the data and possible relationships.

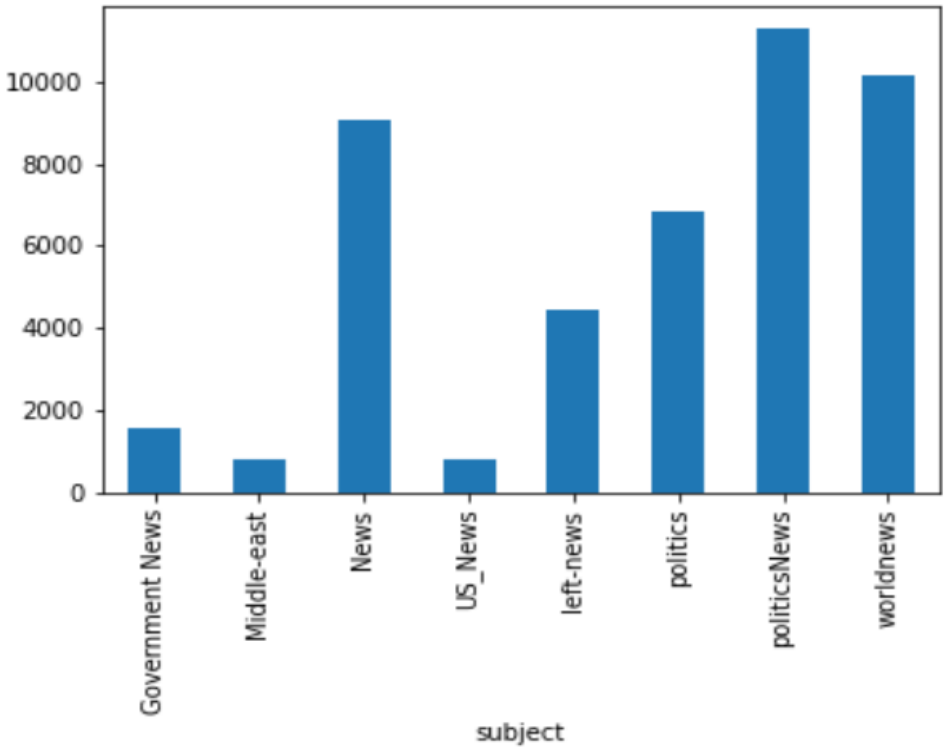


Fig. No. 5.8 Fake News – Articles per subject

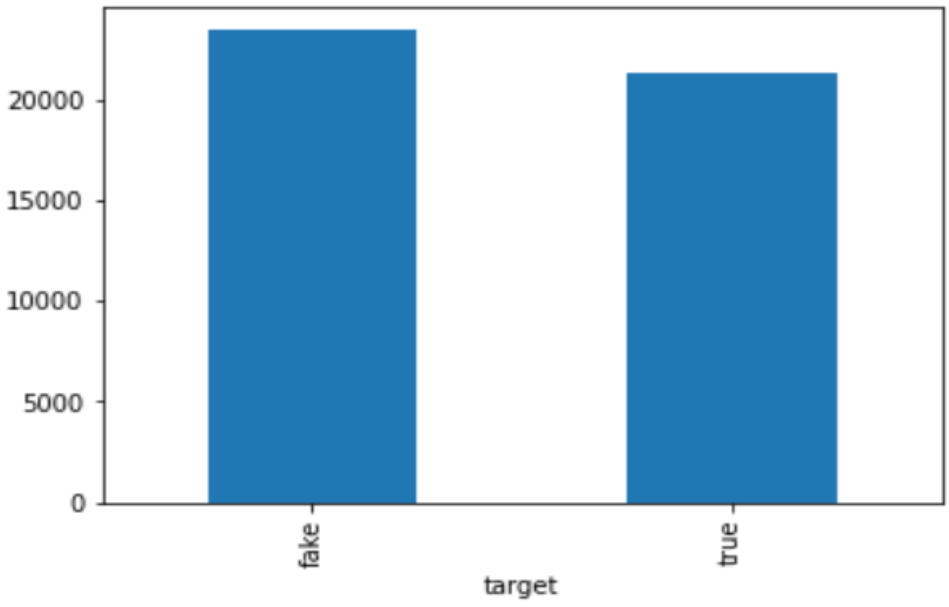


Fig. No. 5.9 Fake news – No. of fake and real articles

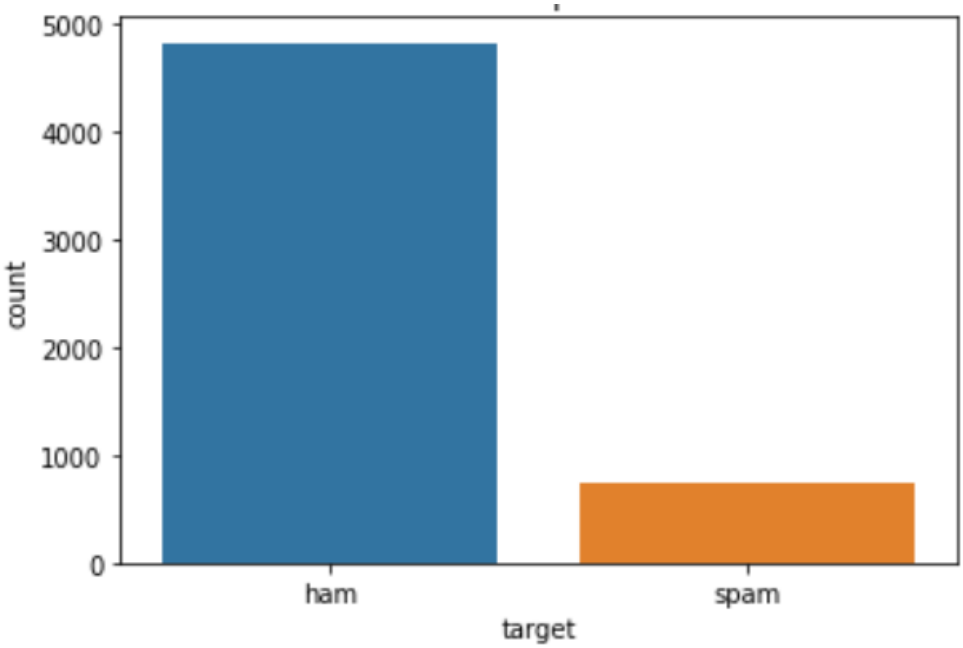


Fig. No. 5.10 Spamming – Distribution of spam and ham

5.2.2.4 FEATURE EXTRACTION

Before fitting the train data and test data into a model. We perform feature extraction to vectorize the data.

1. **BAG OF WORDS**

It is a way of representing text data when modelling text with machine learning algorithms. It describes the occurrence of words within a document. It involves two things:

- **i)** A vocabulary of known words.
- **ii)** A measure of the presence of known words. The model is only concerned with whether known words occur in the document, not where in the document.

Step 1: Collect Data

It was the best of times.

It was the worst of times.

It was the age of wisdom.

It was the age of foolishness

Step 2: Design the Vocabulary Now we can make a list of all of the words in our model vocabulary. The unique words here (ignoring case and punctuation) are:

"it", "was", "the", "best", "of", "times", "worst", "age", "wisdom"

That is a vocabulary of 9 words from a corpus containing 18 words.

Step 3: Create Document Vectors In the first document ("It was the best of times ") and convert it into a binary vector. The simplest scoring method is to mark the presence of words as a Boolean value, 0 for absent, 1 for present. The scoring of the document would look as follows:

"it" = 1 "was" = 1 "the" = 1 "best" = 1 "of"= 1

"times" = 1 "worst" = 0 "age" = 0 "wisdom" = 0

"it was the best of times" = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

2. TF – IDF

TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. It combines two concepts:

i) Term Frequency

ii) Inverse Document Frequency

TEXT 1	i love natural language processing but i hate python
TEXT 2	i like image processing
TEXT 3	i like signal processing and image processing

Table. No. 5.2 TF-IDF Data

Step 1: Create a term frequency matrix where rows are documents and columns are distinct terms throughout all documents. Count word occurrences in every text.

Term frequency = No of Repeatation of words in sentence / No of Words in a sentence

	and	but	hate	i	image	language	like	love	natural	processing	python	signal
Text 1	0	1	1	2	0	1	0	1	1	1	1	0
Text 2	0	0	0	1	1	0	1	0	0	1	0	0
Text 3	1	0	0	1	1	0	1	0	0	2	0	1

Fig. No. 5.11 TF-IDF Term frequency

Step 2: Compute inverse document frequency (IDF) using the previously explained formula

IDF = log (No of sentences / No of sentences containing words)

Term	and	but	hate	i	image	language	like	love	natural	processing	python	signal
IDF	0.47712	0.47712	0.4771	0	0.1760913	0.477121	0.1760913	0.477121	0.47712125	0	0.477121	0.477121

Fig. No. 5.12 TF-IDF Inverse Document Frequency

Step 3: Multiply TF matrix with IDF respectively

	and	but	hate	i	image	language	like	love	natural	processing	python	signal
Text 1	0	0.47712	0.4771	0	0	0.477121	0	0.477121	0.47712125	0	0.477121	0
Text 2	0	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0
Text 3	0.47712	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0.477121

Fig. No. 5.13 TF-IDF matrix

5.2.2.5 MODELLING

In modelling, we are splitting the data as train and test. This is to estimate the performance of machine learning algorithms i.e., used to make predictions on the data not used to train the model. The training data is fit into the model and test data is used to predict. It is a fast and easy procedure that allows us to

compare the performance of machine learning algorithms for predictive-based modelling problems. Supervised Machine Learning Algorithms used here are,

Spamming	**Fake News**	**Bullying/Non-Bullying**
• Naïve Bayes • SVM • KNN • Decision Tree • Random Forest	• Decision Tree • Logistic Regression • Random Forest	• Naïve Bayes • Random Forest • SVM • Decision Tree

Table No. 5.3 Model for spam, fake news, cyber bullying.

CHAPTER - 6

CHAPTER 6

SYSTEM IMPLEMENTATION

CYBER BULLYING

```
#!/usr/bin/env python
# coding: utf-8
# In[104]:
import numpy as np
import pandas as pd
# In[105]:
import os
print(os.listdir("dataset"))
# In[106]:
df = pd.read_json('dataset/dataset.json', lines= True)
# In[107]:
df.head(25)
# In[108]:
df.tail(25)
# In[109]:
df.shape
# In[110]:
df["label"] = df.annotation.apply(lambda x: x.get('label'))
df["label"] = df.label.apply(lambda x: x[0])
df.head()
# In[111]:
df.extras.unique()
# In[112]:
df["notes"] = df.annotation.apply(lambda x: x.get('notes'))
df.notes.unique()
```

```
# In[113]:
df.groupby('label').describe()
# In[114]:
import nltk
nltk.download(['punkt', 'wordnet'])
#Import
import re
import numpy as np
import pandas as pd
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
# In[115]:
def load_data(path):
    df = pd.read_json(path, lines= True)
    df["label"] = df.annotation.apply(lambda x: x.get('label'))
    df["label"] = df.label.apply(lambda x: x[0])
    X = df.content.values
    y = df.label.values
    return X, y
# In[116]:
def tokenize(text):
```

```
    tokens = word_tokenize(text)
lemmatizer = WordNetLemmatizer()
clean_tokens = []
    for tok in tokens:
clean_tok = lemmatizer.lemmatize(tok).lower().strip()
clean_tokens.append(clean_tok)
    return clean_tokens
# In[117]:
def display_results(y_test, y_pred):
    labels = np.unique(y_pred)
confusion_mat = confusion_matrix(y_test, y_pred, labels=labels)
    accuracy = (y_pred == y_test).mean()
print("Labels:", labels)
print("Confusion Matrix:\n", confusion_mat)
print("Accuracy:", accuracy)
# In[118]:
def main():
url = 'dataset/dataset.json'
    X, y = load_data(url)
X_train, X_test, y_train, y_test = train_test_split(X, y)
    print ("X_train: ", len(X_train))
print("X_test: ", len(X_test))
print("y_train: ", len(y_train))
print("y_test: ", len(y_test))
vect = CountVectorizer(tokenizer=tokenize)
tfidf = TfidfTransformer()
    #classfier model
clf = RandomForestClassifier()
```

```
    # train classifier
X_train_counts = vect.fit_transform(X_train)
X_train_tfidf = tfidf.fit_transform(X_train_counts)
clf.fit(X_train_tfidf, y_train)
    # predict on test data
X_test_counts = vect.transform(X_test)
X_test_tfidf = tfidf.transform(X_test_counts)
y_pred = clf.predict(X_test_tfidf)
    # predict on test data
X_test_counts = vect.transform(["whoa stop you stupid sjw"])
X_test_tfidf = tfidf.transform(X_test_counts)
print("\nGiven text: 'whoa stop you stupid sjw' ")
print("Prediction: {}\n".format(clf.predict(X_test_tfidf)))
print("Random Forest")
    # display results
display_results(y_test, y_pred)
mnb = MultinomialNB()
    # train classifier for mnb
X_train_counts = vect.fit_transform(X_train)
X_train_tfidf = tfidf.fit_transform(X_train_counts)
mnb.fit(X_train_tfidf, y_train)
    # predict on test data
X_test_counts = vect.transform(X_test)
X_test_tfidf = tfidf.transform(X_test_counts)
y_pred = mnb.predict(X_test_tfidf)
   # predict on test data
X_test_counts = vect.transform(["whoa stop you stupid sjw"])
X_test_tfidf = tfidf.transform(X_test_counts)
```

```
print("\nGiven text: 'whoa stop you stupid sjw' ")
print("Prediction: {}\n".format(mnb.predict(X_test_tfidf)))
print("Naive Bayes")
    # display results
display_results(y_test, y_pred)
dtc = DecisionTreeClassifier()
   # train classifier
X_train_counts = vect.fit_transform(X_train)
X_train_tfidf = tfidf.fit_transform(X_train_counts)
dtc.fit(X_train_tfidf, y_train)
   # predict on test data
X_test_counts = vect.transform(X_test)
X_test_tfidf = tfidf.transform(X_test_counts)
y_pred = dtc.predict(X_test_tfidf)
   # predict on test data
X_test_counts = vect.transform(["whoa stop you stupid sjw"])
X_test_tfidf = tfidf.transform(X_test_counts)
print("\nGiven text: 'whoa stop you stupid sjw' ")
print("Prediction: {}\n".format(dtc.predict(X_test_tfidf)))
print("Decision Tree")
    # display results
display_results(y_test, y_pred)
svm = SVC()
   # train classifier
X_train_counts = vect.fit_transform(X_train)
X_train_tfidf = tfidf.fit_transform(X_train_counts)
svm.fit(X_train_tfidf, y_train)
   # predict on test data
```

```
X_test_counts = vect.transform(X_test)
X_test_tfidf = tfidf.transform(X_test_counts)
y_pred = svm.predict(X_test_tfidf)
  # predict on test data
X_test_counts = vect.transform(["whoa stop you stupid sjw"])
X_test_tfidf = tfidf.transform(X_test_counts)
print("\nGiven text: 'whoa stop you stupid sjw' ")
print("Prediction: {}\n".format(svm.predict(X_test_tfidf)))
   print("SVM")
  # display results
display_results(y_test, y_pred)
main()
```

SPAMMING

```
#!/usr/bin/env python
# coding: utf-8
# In[1]:
import numpy as np
import pandas as pd
import os
# In[2]:
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split
```

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, confusion_matrix
# In[3]:
##Datasets
# In[4]:
data = pd.read_csv("spam.csv", encoding = 'latin1')
data.head()
# In[5]:
# drop unavailable attributes
data = data.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis = 1)
# In[6]:
# rename columns
data = data.rename(columns ={"v1":"target", "v2":"text"})
data.head()
# In[7]:
#for counting ham and spam
data.target.value_counts()
# In[8]:
```

```
data.groupby('target').describe()
# In[9]:
#plotting graph for distribution
sns.countplot(x = "target", data = data)
data.loc[:, 'target'].value_counts()
plt.title('Distribution of Spam and Ham')
# In[10]:
# plotting graph by length.
ham =data[data['target'] == 'ham']['text'].str.len()
sns.distplot(ham, label='Ham')
spam = data[data['target'] == 'spam']['text'].str.len()
sns.distplot(spam, label='Spam')
plt.title('Distribution by Length')
plt.legend()
# In[11]:
#plotting graph by digits.
ham1 = data[data['target'] == 'ham']['text'].str.replace(r'\D+', '').str.len()
sns.distplot(ham1, label='Ham')
spam1 = data[data['target'] == 'spam']['text'].str.replace(r'\D+', '').str.len()
sns.distplot(spam1, label='Spam')
plt.title('Distribution by Digits')
plt.legend()
# In[12]:
#plotting graph for non-digits.
ham2 = data[data['target'] == 'ham']['text'].str.replace(r'\w+', '').str.len()
sns.distplot(ham2, label='Ham')
spam2 = data[data['target'] == 'spam']['text'].str.replace(r'\w+', '').str.len()
sns.distplot(spam2, label='Spam')
```

```
plt.title('Distribution of Non-Digits')
plt.legend()
# In[13]:
#for counting frequently occurence of spam and ham.
count1 = Counter("
".join(data[data['target']=='ham']["text"]).split()).most_common(30)
data1 = pd.DataFrame.from_dict(count1)
data1 = data1.rename(columns={0: "words of ham", 1 : "count"})
count2 = Counter("
".join(data[data['target']=='spam']["text"]).split()).most_common(30)
data2 = pd.DataFrame.from_dict(count2)
data2 = data2.rename(columns={0: "words of spam", 1 : "count_"})
# In[14]:
data1.plot.bar(legend = False, color = 'purple',figsize = (20,15))
y_pos = np.arange(len(data1["words of ham"]))
plt.xticks(y_pos, data1["words of ham"])
plt.title('Top 30 words of ham')
plt.xlabel('words')
plt.ylabel('number')
plt.show()
# In[15]:
data2.plot.bar(legend = False, color = 'green', figsize = (20,17))
y_pos = np.arange(len(data2["words of spam"]))
plt.xticks(y_pos, data2["words of spam"])
plt.title('Top 30 words of spam')
plt.xlabel('words')
plt.ylabel('number')
plt.show()
```

```
# In[16]:
# Function to plot the confusion matrix (code from https://scikit-
learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)
from sklearn import metrics
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
cmap=plt.cm.Blues):
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
print("Normalized confusion matrix")
    else:
print('Confusion matrix, without normalization')
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
plt.text(j, i, cm[i, j],
horizontalalignment="center",
color="white" if cm[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
# In[17]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['target'],
test_size = 0.3, random_state = 37)
print ("X_train: ", len(X_train))
print("X_test: ", len(X_test))
print("y_train: ", len(y_train))
print("y_test: ", len(y_test))
# In[18]:
corpus = []
for i in range(0, 5572):
    review = re.sub('[^a-zA-Z]', ' ', data['text'][i])
    review = review.lower()
    review = review.split()
ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in
set(stopwords.words('english'))]
    review = ' '.join(review)
corpus.append(review)
# In[19]:
cv = CountVectorizer(max_features = 1500)
cv.fit(X_train)
# In[20]:
X_train_cv = cv.transform(X_train)
X_train_cv
# In[21]:
X_test_cv = cv.transform(X_test)
X_test_cv
```

```
# In[22]:
mnb = MultinomialNB(alpha = 0.5)
mnb.fit(X_train_cv,y_train)
y_mnb = mnb.predict(X_test_cv)
# In[23]:
print('Naive Bayes Accuracy: ', accuracy_score( y_mnb , y_test))
print('Naive Bayes confusion_matrix: ', confusion_matrix(y_mnb, y_test))
cm = metrics.confusion_matrix(y_test, y_mnb)
plot_confusion_matrix(cm, classes=['spam', 'ham'])
# In[24]:
svc = SVC(kernel='sigmoid', gamma=1.0)
svc.fit(X_train_cv,y_train)
y_svc = svc.predict(X_test_cv)
# In[25]:
print('SVM Accuracy: ', accuracy_score( y_svc , y_test))
print('SVM confusion_matrix: ', confusion_matrix(y_svc, y_test))
cm = metrics.confusion_matrix(y_test, y_svc)
plot_confusion_matrix(cm, classes=['spam', 'ham'])
# In[26]:
knc = KNeighborsClassifier(n_neighbors=100)
knc.fit(X_train_cv,y_train)
y_knc = knc.predict(X_test_cv)
# In[27]:
print('KNeighborsAccuracy_score: ',accuracy_score(y_test,y_knc))
print('KNeighborsconfusion_matrix: ', confusion_matrix(y_test, y_knc))
cm = metrics.confusion_matrix(y_test, y_knc)
plot_confusion_matrix(cm, classes=['spam', 'ham'])
# In[28]:
```

```
dtc = DecisionTreeClassifier(min_samples_split=7, random_state=252)
dtc.fit(X_train_cv,y_train)
y_dtc = dtc.predict(X_test_cv)
# In[29]:
print('Decision Tree Accuracy: ',accuracy_score(y_test,y_dtc))
print('Decision Tree confusion_matrix: ', confusion_matrix(y_dtc, y_test))
cm = metrics.confusion_matrix(y_test, y_dtc)
plot_confusion_matrix(cm, classes=['spam', 'ham'])
# In[30]:
rfc = RandomForestClassifier(n_estimators=37, random_state=252)
rfc.fit(X_train_cv,y_train)
y_rfc = rfc.predict(X_test_cv)
print('Random Forest Accuracy_score: ',accuracy_score(y_test,y_rfc))
print('Random Forest confusion_matrix: ', confusion_matrix(y_rfc, y_test))
cm = metrics.confusion_matrix(y_test, y_rfc)
plot_confusion_matrix(cm, classes=['spam', 'ham'])
```

FAKE NEWS

```
#!/usr/bin/env python
# coding: utf-8
# # Fake news detection
# In[1]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
```

```
from sklearn.feature_extraction.text import TfidfTransformer
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.pipeline import Pipeline
# ## Read datasets
# In[2]:
fake = pd.read_csv("data/Fake.csv")
true = pd.read_csv("data/True.csv")
# In[3]:
fake.shape
# In[4]:
true.shape
# ## Data cleaning and preparation
# In[5]:
# Add flag to track fake and real
fake['target'] = 'fake'
true['target'] = 'true'
# In[6]:
# Concatenate dataframes
data = pd.concat([fake, true]).reset_index(drop = True)
data.shape
# In[7]:
# Shuffle the data
from sklearn.utils import shuffle
data = shuffle(data)
data = data.reset_index(drop=True)
```

```
# In[8]:
# Check the data
data.head()
# In[9]:
# Removing the date (we won't use it for the analysis)
data.drop(["date"],axis=1,inplace=True)
data.head()
# In[10]:
# Removing the title (we will only use the text)
data.drop(["title"],axis=1,inplace=True)
data.head()
# In[11]:
# Convert to lowercase
data['text'] = data['text'].apply(lambda x: x.lower())
data.head()
# In[12]:
# Remove punctuation
import string
def punctuation_removal(text):
all_list = [char for char in text if char not in string.punctuation]
clean_str = ''.join(all_list)
    return clean_str
data['text'] = data['text'].apply(punctuation_removal)
# In[13]:
# Check
data.head()
# In[14]:
# Removing stopwords
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if
word not in (stop)]))
# In[15]:
data.head()
# ## Basic data exploration
# In[16]:
# How many articles per subject?
print(data.groupby(['subject'])['text'].count())
data.groupby(['subject'])['text'].count().plot(kind="bar")
plt.show()
# In[17]:
# How many fake and real articles?
print(data.groupby(['target'])['text'].count())
data.groupby(['target'])['text'].count().plot(kind="bar")
plt.show()
# In[18]:
# Word cloud for fake news
from wordcloud import WordCloud
fake_data = data[data["target"] == "fake"]
all_words = ' '.join([text for text in fake_data.text])
wordcloud = WordCloud(width= 800, height= 500,
max_font_size = 110,
                      collocations = False).generate(all_words)
plt.figure(figsize=(10,7))
```

```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
# In[19]:
# Word cloud for real news
from wordcloud import WordCloud
real_data = data[data["target"] == "true"]
all_words = ' '.join([text for text in fake_data.text])
wordcloud = WordCloud(width= 800, height= 500,
max_font_size = 110,
                      collocations = False).generate(all_words)
plt.figure(figsize=(10,7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
# In[20]:
from nltk import tokenize
token_space = tokenize.WhitespaceTokenizer()
def counter(text, column_text, quantity):
all_words = ' '.join([text for text in text[column_text]])
token_phrase = token_space.tokenize(all_words)
  frequency = nltk.FreqDist(token_phrase)
df_frequency = pd.DataFrame({"Word": list(frequency.keys()),
                             "Frequency": list(frequency.values())})
df_frequency = df_frequency.nlargest(columns = "Frequency", n = quantity)
plt.figure(figsize=(12,8))
ax = sns.barplot(data = df_frequency, x = "Word", y = "Frequency", color =
'blue')
```

```
ax.set(ylabel = "Count")
plt.xticks(rotation='vertical')
plt.show()
# In[21]:
# Most frequent words in fake news
counter(data[data["target"] == "fake"], "text", 20)
# In[22]:
# Most frequent words in real news
counter(data[data["target"] == "true"], "text", 20)
# ## Modeling
# In[23]:
# Function to plot the confusion matrix (code from https://scikit-
learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)
from sklearn import metrics
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
cmap=plt.cm.Blues):
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
print("Normalized confusion matrix")
```

```
    else:
print('Confusion matrix, without normalization')
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
plt.text(j, i, cm[i, j],
horizontalalignment="center",
color="white" if cm[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
# ### Peparing the data
# In[24]:
# Split the data
X_train,X_test,y_train,y_test = train_test_split(data['text'], data.target,
test_size=0.2, random_state=42)
print ("X_train: ", len(X_train))
print("X_test: ", len(X_test))
print("y_train: ", len(y_train))
print("y_test: ", len(y_test))
# ### Logistic regression
# In[25]:
# Vectorizing and applying TF-IDF
from sklearn.linear_model import LogisticRegression
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LogisticRegression())])
# Fitting the model
model = pipe.fit(X_train, y_train)
```

```
# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test,
prediction)*100,2)))
# In[26]:
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Real', 'Fake'])
# ### Decision Tree Classifier
# In[27]:
from sklearn.tree import DecisionTreeClassifier
# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', DecisionTreeClassifier(criterion= 'entropy',
max_depth = 20,
                                    splitter='best',
random_state=42))])
# Fitting the model
model = pipe.fit(X_train, y_train)
# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test,
prediction)*100,2)))
# In[28]:
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
# ### Random Forest Classifier
#In[29]:
```

```
from sklearn.ensemble import RandomForestClassifier
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', RandomForestClassifier(n_estimators=50,
criterion="entropy"))])
model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test,
prediction)*100,2)))
# In[30]:
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

CHAPTER - 7

CHAPTER 7

PERFORMANCE ANALYSIS

Evaluation metrics are used to measure the quality of the machine learning model. Evaluating machine learning models or algorithms is essential for any project. By using evaluation metrics, we can ensure that the model is operating correctly and optimally.

It is very important to use multiple evaluation metrics to evaluate your model. This is because a model may perform well using one measurement from one evaluation metric, but may perform poorly using another measurement from another evaluation metric. There are many different types of evaluation metrics available to test a model. They are confusion matrix, accuracy, precision, recall, F1-score,false positive rate, Receiver operator characteristics curve (ROC), Precision- Recall (PR) curve, Logarithmic loss, mean absolute error and root mean squared error.

In this project, we include confusion matrix and classification accuracy.

A confusion matrix gives us a matrix as output and describes the complete performance of the model.

Classification of a test dataset produces four outcomes – true positive, false positive, true negative, and false negative.

True positive (TP): correct positive prediction

False positive (FP): incorrect positive prediction

True negative (TN): correct negative prediction

False negative (FN): incorrect negative prediction

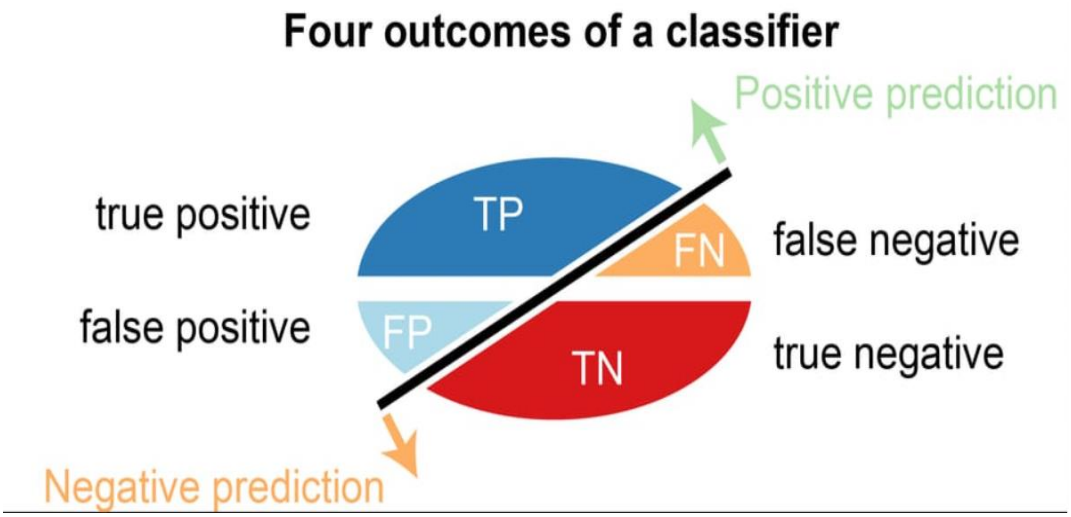


Fig. No. 7.1 Confusion matrix – four outcomes

Classification accuracy is the ratio of the number of correct predictions to the total number of input samples, which is usually what we refer to when we use the term accuracy. The best accuracy is 1.0, whereas the worst is 0.0. It can also be calculated by 1 – ERR.

Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

ACCURACY = (TP + TN) / (TP + TN + FN + FP)

ACCURACY = (TP + TN) / (P + N)

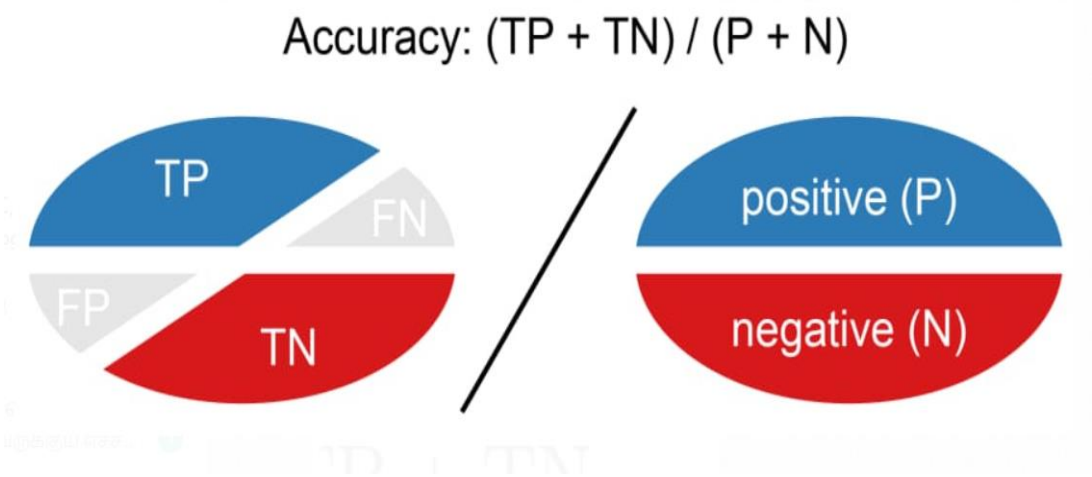


Fig. No. 7.2 Accuracy Calculation

Here the evaluation metrics predicted are shown below:

7.1CONFUSION MATRIX

7.1.1 FAKE NEWS

LOGISTIC REGRESSION

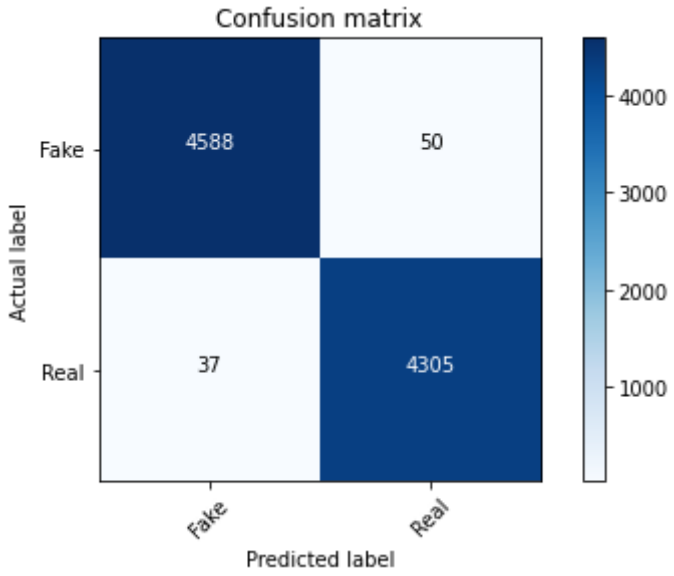


Fig No. 7.3 LR – Fake News Confusion Matrix

True positive = 4588 predicted values correctly predicted as actual positive

False positive = 37 negative values predicted as positive

False negative = 50 positive values predicted as negative

True negative = 4304 predicted values correctly predicted as an actual negative

DECISION TREE CLASSIFIER

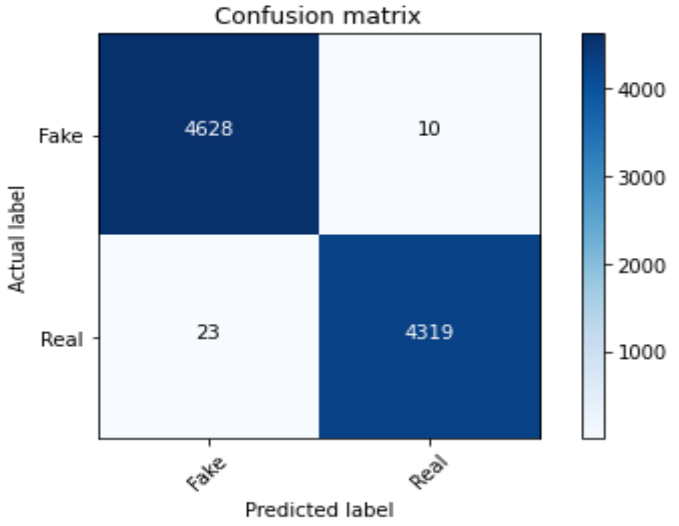


Fig No. 7.4 DT – Fake News Confusion Matrix

True positive = 4628 predicted values correctly predicted as actual positive

False positive = 23 negative values predicted as positive

False negative = 10 positive values predicted as negative

True negative = 4319 predicted values correctly predicted as an actual negative

RANDOM FOREST CLASSIFIER

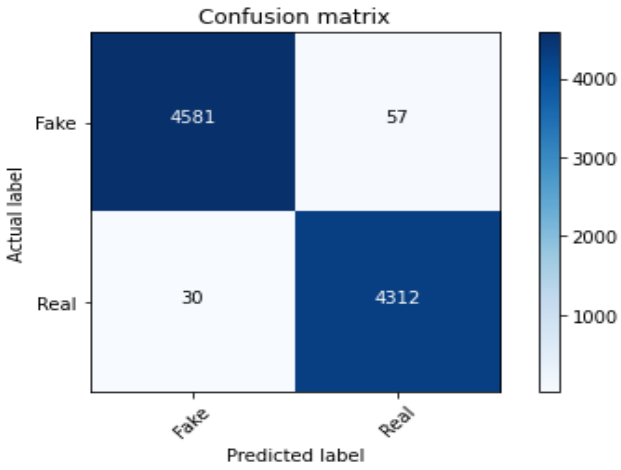


Fig. No. 7.5 RF -Fake News Confusion Matrix

True positive = 4581 predicted values correctly predicted as actual positive

False positive = 30 negative values predicted as positive

False negative = 57 positive values predicted as negative

True negative = 4312 predicted values correctly predicted as an actual negative

7.1.2 SPAMMING

NAÏVE BAYES

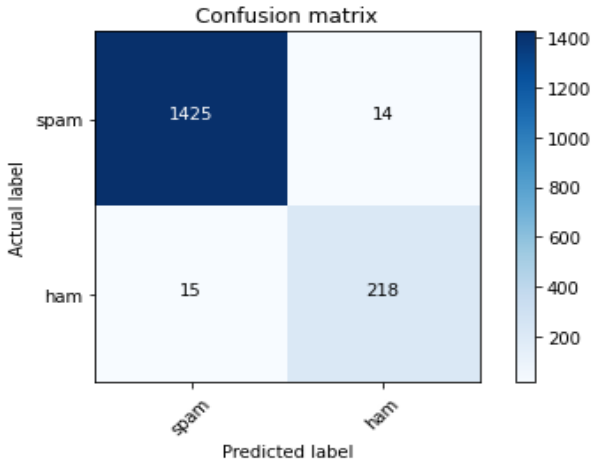


Fig. No. 7.6 NB – Spamming Confusion Matrix

True positive = 1425 predicted values correctly predicted as actual positive

False positive = 15 negative values predicted as positive

False negative = 14 positive values predicted as negative

True negative = 218 predicted values correctly predicted as an actual negative

SUPPORT VECTOR MACHINE

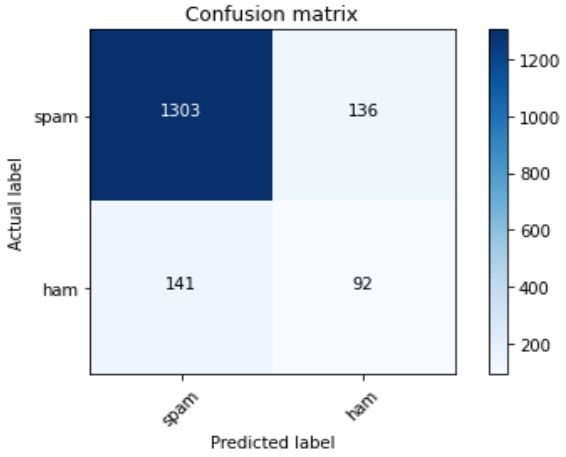


Fig. No. 7.7 SVM – Spamming Confusion Matrix

True positive = 1303 predicted values correctly predicted as actual positive

False positive = 141 negative values predicted as positive

False negative = 136 positive values predicted as negative

True negative = 92 predicted values correctly predicted as an actual negative

KNN

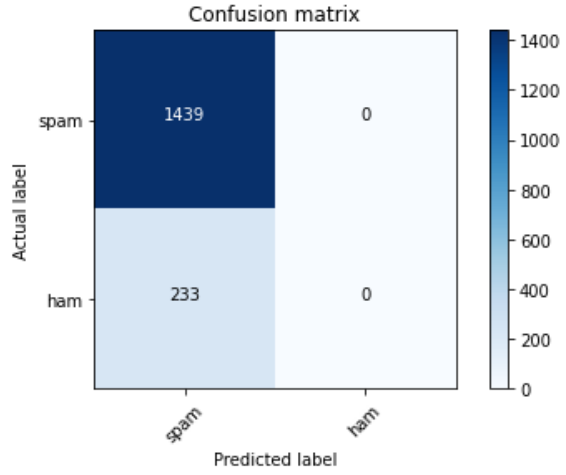


Fig. No. 7.8 KNN – Spamming Confusion Matrix

True positive = 1439 predicted values correctly predicted as actual positive

False positive = 233 negative values predicted as positive

False negative = 0 positive values predicted as negative

True negative = 0 predicted values correctly predicted as an actual negative

DECISION TREE

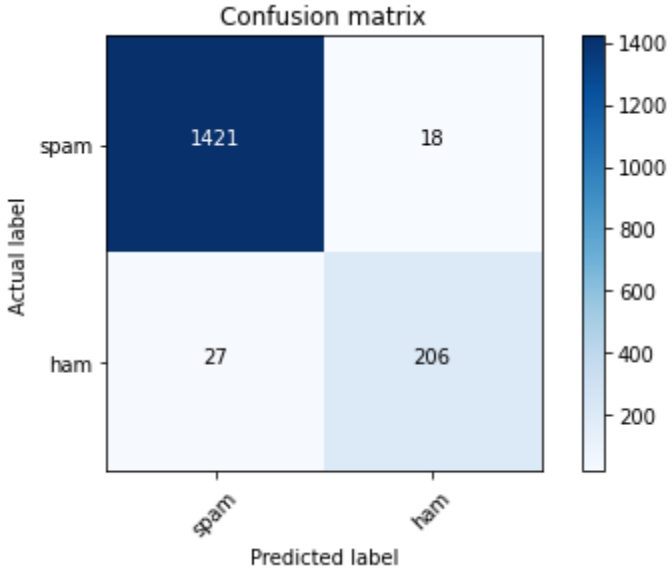


Fig. No. 7.9 DT – Spamming Confusion Matrix

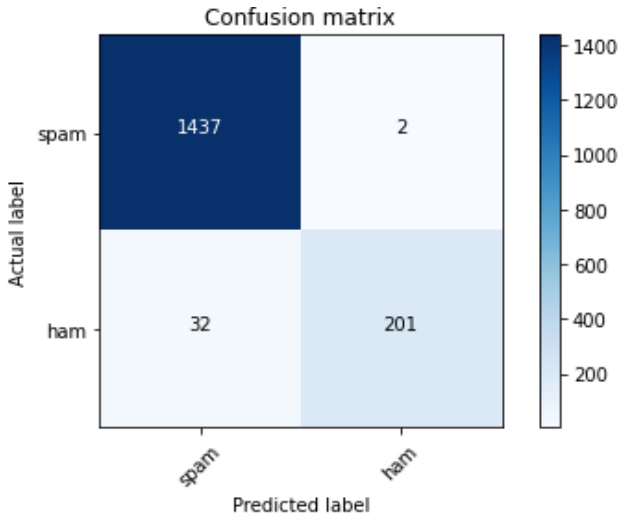
True positive = 1421 predicted values correctly predicted as actual positive

False positive = 27 negative values predicted as positive

False negative = 18 positive values predicted as negative

True negative = 206 predicted values correctly predicted as an actual negative

RANDOM FOREST



Fig, No 7.10 RF – Spamming Confusion Matrix

True positive = 1437 predicted values correctly predicted as actual positive

False positive = 32 negative values predicted as positive

False negative = 2 positive values predicted as negative

True negative = 201 predicted values correctly predicted as an actual negative

7.2 ACCURACY

7.2.1 FAKE NEWS

FAKE NEWS	
Model	**Accuracy**
Decision Tree	99.54%
Logistic Regression	98.74%
Random Forest	99.12%

Table No. 7.1 Fake News Performance Analysis

From the above table the Decision Tree model has highest accuracy as 94.62%. As a result, this machine learning model works efficiently for detecting Fake News.

7.2.2 SPAMMING

SPAMMING	
Model	**Accuracy**
Naïve Bayes	98.26%
SVM	83.43%
KNN	86.06%
Decision Tree	97.30%
Random Forest	97.96%

Table No. 7.2 Spamming Performance Analysis

From the above table, the Naïve Bayes model has highest accuracy as 98.26%. As a result, this machine learning model works efficiently for detecting spam.

7.2.3 CYBER BULLYING

BULLYING& NON-BULLYING	
Model	**Accuracy**
Decision Tree	85.82%
Naïve Bayes	76.58%
SVM	90.96%
Random Forest	95.24%

Table No. 7.3 Cyber Bullying Performance Analysis

From the above table, the Random Forest model has highest accuracy as 95.24%. As a result, this machine learning model works efficiently for detecting Cyberbullying activities.

CHAPTER - 8

CHAPTER 8

CONCLUSION

8.1CONCLUSIONANDFUTURE ENCHANTMENTS

False information and bullying on social media have become serious issues in recent years, and a system that can detect such texts would undoubtedly be beneficial. As a result, text analysis is performed to identify inconsistencies in the text, which is then subjected to various supervised machine learning algorithms to determine the most precise estimation to resolve the issues. This way, we can assist people in making more informed decisions, as well as ensuring that they are not duped into believing what others want them to believe. Bullying, fake news, and spamming can all be mitigated as a result of this.

Testing and training corpora can be tailored to a specific domain in the future, as corpus vocabulary varies by domain.

APPENDICES

A.1 SAMPLE SCREENS

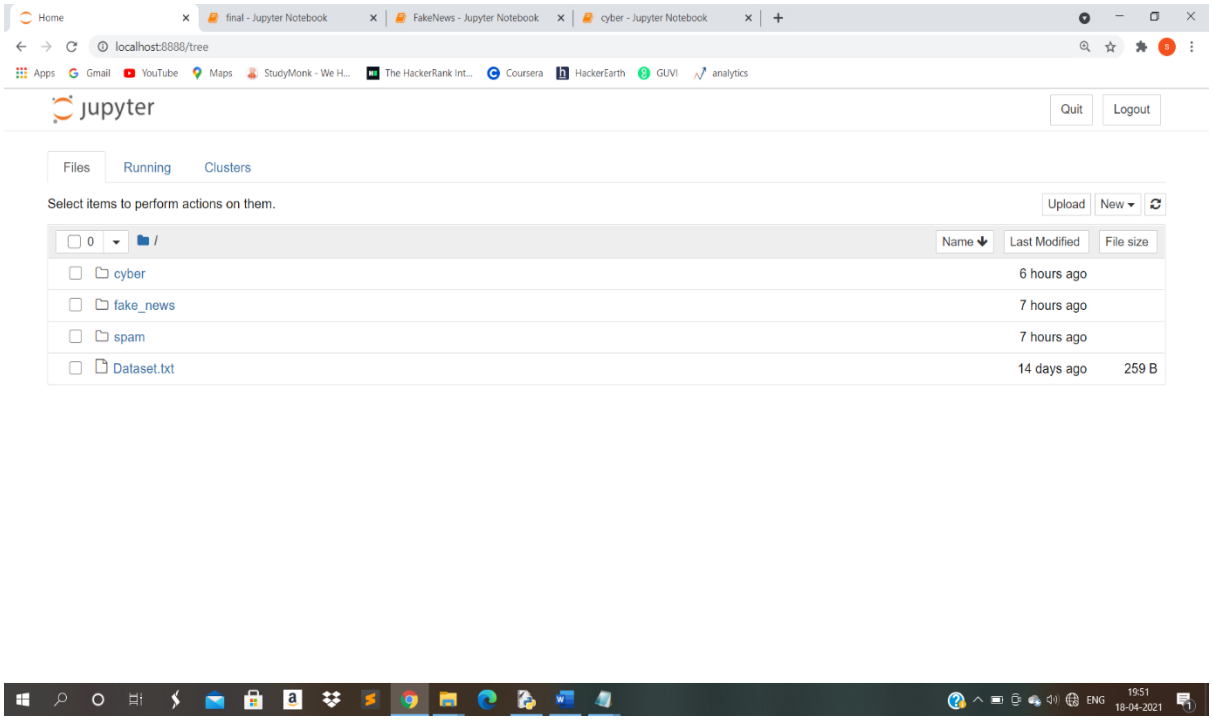


Fig No. A.1 Coding Folders

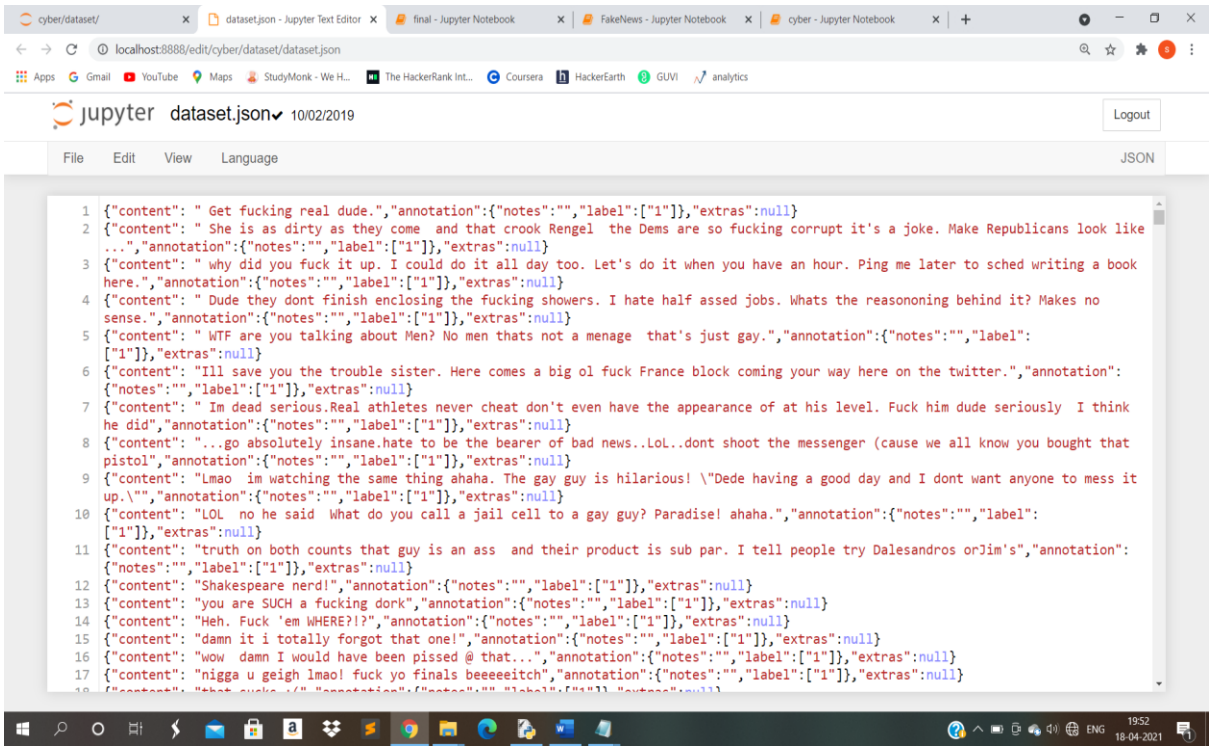


Fig. No. A.2 Dataset

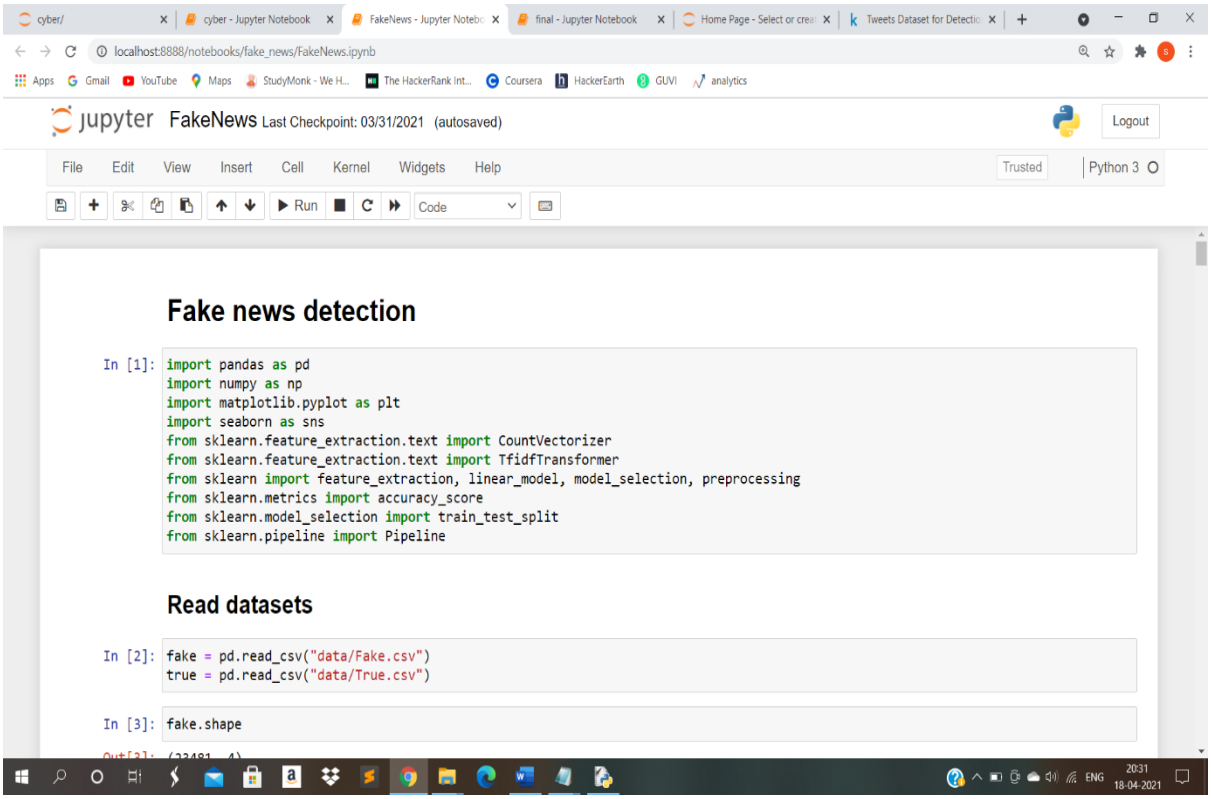


Fig. No. A.3 Code sample

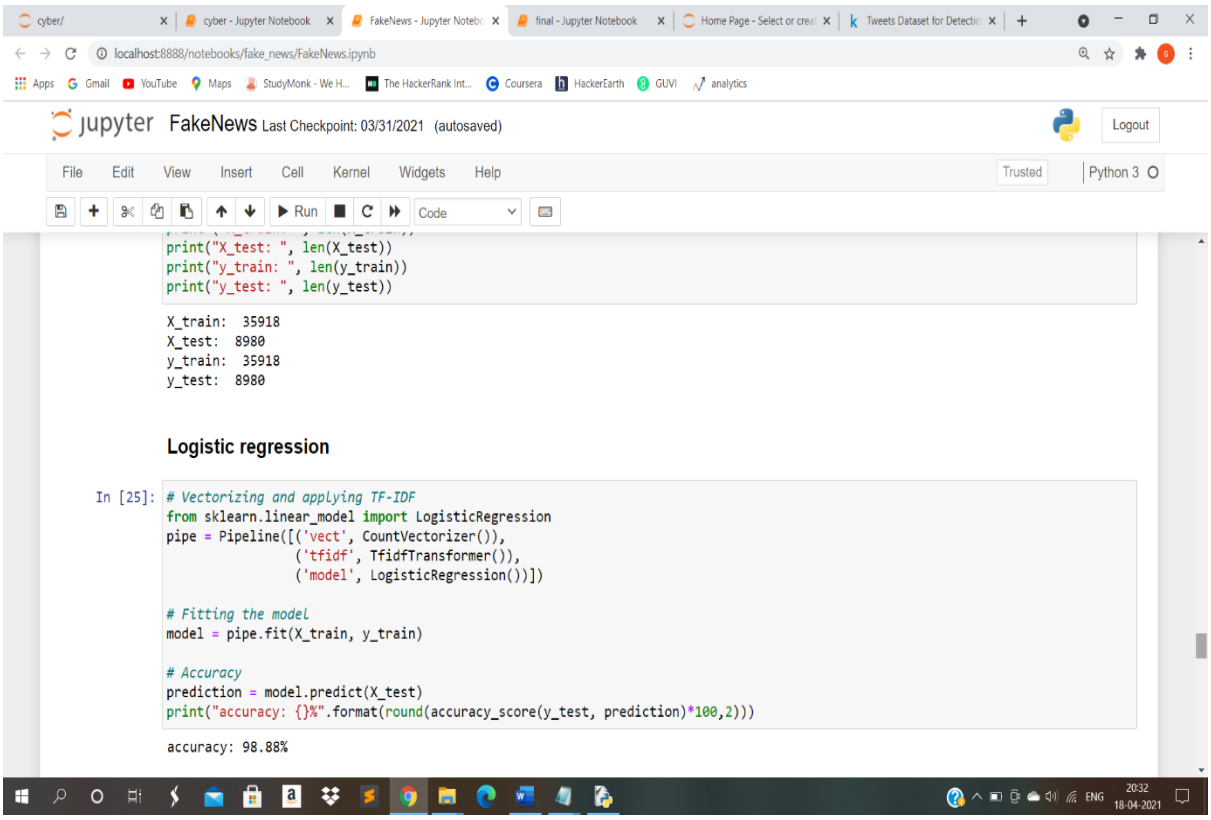


Fig. No. A.4 Fake News Output Sample

A.2 PUBLICATIONS

- "FINDING INCONSISTENCY OF SECURITY INFORMATION FROM UNSTRUCTURED TEXT" is been published under International Research Journal of Engineering and Technology (IRJET) **(https://www.irjet.net/).**
- IRJET is a peer-reviewed journal which focus on current trends of research.
- IRJET Volume 8, Issue 4, April 2021 https://www.irjet.net/volume8-issue4 (S.No: 302)
- Manuscript - https://www.irjet.net/archives/V8/i4/IRJET-V8I4302.pdf


e-ISSN: 2395-0056 p-ISSN: 2395-0072
International Research Journal of Engineering and Technology (IRJET)
(An ISO 9001 : 2008 Certified Journal)
Is hereby awarding this certificate to
Sanggami R
In recognition the publication of the manuscript entitled
Finding Inconsistency of Security Information from Unstructured Text
published in our Journal Volume 8 Issue 4 April 2021
Editor in Chief
E-mail : editor@irjet.net
Impact Factor : 7.529
www.irjet.net


e-ISSN: 2395-0056 p-ISSN: 2395-0072
International Research Journal of Engineering and Technology (IRJET)
(An ISO 9001 : 2008 Certified Journal)
Is hereby awarding this certificate to
Saranya R
In recognition the publication of the manuscript entitled
Finding Inconsistency of Security Information from Unstructured Text
published in our Journal Volume 8 Issue 4 April 2021
Editor in Chief
E-mail : editor@irjet.net
Impact Factor : 7.529
www.irjet.net

InternationalResearchJournalofEngineeringandTechnology(IRJET) e-ISSN:2395-0056
Volume:08Issue:04|Apr2021 www.irjet.net p-ISSN:2395-0072

FindingInconsistencyofSecurityInformationfromUnstructuredText

PunithaG,SanggamiR,SaranyaR, K.Sangeetha

Student, Dept of Computer Science Engineering, PEC College, Tamil Nadu, India
Student, Dept of Computer Science Engineering, PEC College, Tamil Nadu, India
Student,DeptofComputer ScienceEngineering,PECCollege,TamilNadu,India
AssociateProfessor,DeptofComputerScienceEngineering,PECCollege,TamilNadu,India

---

Abstract - Open source intelligence textual data has become a big topic in a variety of fields, including industry, law enforcement, military, and cyber security. Millions of people use social networking sites all over the world. The user's experiences with social media platforms like Twitter and Face book have a huge effect on their everyday lives, with some detrimental effects. The identification of scams and phony Social media users has become a common research area in the field of modern online social networks (OSNs). We show that Supervised Machine learning can overcome all the bottlenecks. We here find the conflict, disagreement or mismatch of the secured information from the unstructured data. Text analyses and classification is made to detect the Cyber bullying activities, Fake News Identification and Spamming.

Key Words*: *Machine Learning, Cyber bullying, Fake News,Spamming,OnlineSocialNetworks.

1. INTRODUCTION

One of the most effective and efficient ways to communicate today is through the Internet and social media. It has made it extremely simple for anyone to publish content on their website, blog, or social media profile and potentially reach large audiences. The internet allows us to connect with a wide range of people and read news and information from all over the world, whether through Face book, Twitter, Yahoo, or any other website. Because there is such a high demand for resources, online networking sites have evolved into a source of real-time data about any user. As a result, it provides commentary on a previously unseen method of information discrimination. Fraudsters can easily deceive many people who have little knowledge of online networking. Although the Internet has many positive aspects and can be extremely beneficial, it can also be a source of danger. With the growth of Online networking sites, there needs to be analysis and study of user behavior. As the data available and shared on the internet can be in the form of text, video, audio, and so on. We here focus only on the text. One of the most crucial aspects of text analysis is text classification. Since text can be a very rich source of knowledge, but due to its unstructured nature, extracting insights from it can be difficult and time-consuming. So, we assign tags or categories to text according to its content, helping structured and analyze the text, quickly and cost-effectively, to automate processes and enhance data-driven decisions. Here text analysis is done to detect Cyber bullying activities, Fake news (False Information), and spamming. Many people get their news from social media sites and networks, and determining whether or how reports are accurate can be challenging. Information overload and a general lack of understanding about how the internet works by people have also contributed to an increase in fake news, Bullying, and spamming activities. Sending, publishing, or sharing negative, damaging, misleading, or mean material about someone else is considered cyber bullying. It can also include sharing personal or private information about someone causing embarrassment or humiliation. This crosses the line into unlawful or criminal behavior. False information is news, reports, or hoaxes that are intended to mislead or misinform readers. These stories are usually created to either influence or confuse people, and they can be a lucrative business for online publishers. Spamming is the practice of sending unsolicited or misleading messages across the Internet. It is often used for commercial advertising. All these issues can be resolved by text classification and analysis are made and detection is performed. In this, real-time unstructured data later modified to a structured format is collected from Kaggle. Then, preprocessing of the text is made by bringing it into the corpus (collection of data), and then vectorize the corpus by feature extraction. With the help of a supervised machine learning model, the classification and analysis are made. Then the accurate prediction for each model is determined. By doing this we can predict which model will give a higher accuracy rate. The model with high accuracy can help to detect fake news, spamming, and cyber bullying activities effectively. Maybe this will save a life or reduce the number of teens touched by cyber-bullying activities, fake news, Spamming, or any other internet dangers.

2. RELATEDWORKS

The following sub-sections discuss some of the research that has been done by researchers in the areas of cyber bullying, fake news, and spam detection.

2.1 Gender and Age Detection in Cyber bullying Texts Using Computational Stylometry and Machine Learning

The purpose of this paper is to demonstrate the value of Computational Stylometry (CS) and Machine Learning (ML) in detecting author gender and age in cyber bullying texts. We created a cyber bullying detection platform and show the results of gender and age detection in cyber bullying texts we collected in terms of Precision, Recall, and F-Measure.

The benefits are as follows:

1. We can categorize and analyze cyber bullying text by grouping it into taxonomy (linguistic rules) to detect cyber bullies' gender and age.

The disadvantages include:

1. Focusing on categories based solely on written text
2. Developing a smaller number of stylistic features.

2.2 An integrated approach for malicious tweets detection using NLP

Detection of malicious user accounts has been the focus of many previous studies. Detecting spam or spammers on Twitter is a relatively new area of social network research. However, we present a method based on two new aspects: the detection of spam-tweets without knowing the user's previous background, and the other based on language analysis for detecting spam on Twitter in topics that are currently trending. Topics of discussion that are popular at the time are known as trending topics. Spammers benefit from the growing micro blogging phenomenon. Our research uses language tools to detect spam tweets. We began by gathering tweets related to a variety of popular topics and categorizing them as malicious or non-malicious. We extracted a number of features based on language models using language as a tool after a labeling process. We also assess performance and categorize tweets as spam or not. As a result, our system can be used to detect spam on Twitter, with a focus on tweet analysis rather than user accounts.

The merits are as follows:

1. We detect spam accounts using the SVM algorithm, which yields a standard result.

The Demerits, on the other hand, are

1. It can only be used to detect spam on Twitter, with a focus on tweet detection.

2.3 Unsupervised cyber bullying detection in social networks

Modern young people (also known as "digital natives") have grown up in a world dominated by new technologies, in which communication is pushed to near-real-time levels and there are no boundaries to forming relationships with other people or communities. However, due to the rapid pace of evolution, young people are unable to distinguish between consciously acceptable and potentially harmful behaviors, and a new phenomenon known as cyber bullying is gaining traction, attracting the attention of educators and the media. “Willful and repeated harm inflicted through the use of electronic devices” is what cyber bullying is defined as. Using techniques derived from NLP (Natural Language Processing) and machine learning, we propose a possible solution for automatic detection of bully traces across a social network in this paper. We will create a model based on Growing Hierarchical SOMs that can efficiently cluster documents containing bully traces and is based on semantic and syntactic features of textual sentences. We fine-tuned our model to work with Twitter, but we also put it to the test with other social media platforms like YouTube and Form spring. Finally, we present our findings, demonstrating that the proposed unsupervised approach can be used effectively in some scenarios with good results.

The merits are as follows:

1. Use of an unsupervised approach to analyze cyber bullying with several handcrafted features that were used to catch the cyber bullies' semantic and syntactic communication behavior.

The disadvantages are as follows:

1. Less accuracy and performance results.

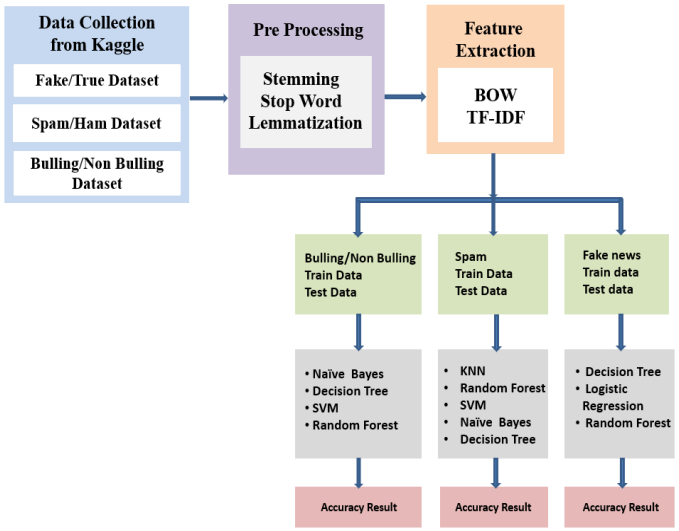
3. PROPOSEDSYSTEM

Fake news on the Internet and cyber bullying on social media are major concerns for everyone in society, including the government, policymakers, organizations, businesses, and citizens. Fake news spreading on social media has become a significant problem in this world, with the potential to lead to mob violence. Bullying can have a negative impact on a person's physical, mental, and emotional well-being, leading to depression. As a result, a recipient of spam emails risk having their computers infected with a malicious program. Spam will clog mail servers, slowing down all email and putting a strain on the Server. Because spam email, fake news detection, and cyber bullying activities all fall under the category of text analysis, we've combined all of these domains and developed a system that can detect such texts. Using a supervised machine learning algorithm, we perform text classification on the dataset to identify fake/true news, spam/ham in the mail, bullying and non-bullying activity in our proposed system. We clean and prepare the real-time dataset before applying it directly to the algorithm, using pre-processing techniques such as stemming, lemmatization, and feature extraction to derive values (features in vector format) that are intended to be informative and non-redundant. When features are applied to various supervised

Machine learning models, the accuracy of each model is predicted, and the model with the highest accuracy is most effective in detecting the problem.

4. SYSTEMARCHITECTURE

When features are applied to various supervised machine learning models, the accuracy of each model is predicted, and the model with the highest accuracy is most effective in detecting the problem.



Real-time Unstructured data is made into structured format are collected from Kaggle. We are collecting datasets specifically for Bullying, Fake News, and Spamming. Then the datasets collected will be subjected to preprocessing like Stemming, Stop Word and tokenization if needed lemmatization. To analyze preprocessed data, it must first be converted into features, or the text must be converted into numerical vectors. A technique for extracting features from text is by BOW and TF - IDF. A bag of words is a method to represent text into numerical features. Bag of Features can be easily created by using the count Vectorizier function. This process is called feature extraction (or Vectorization). It is used to convert a collection of text documents to a vector of token counts. The TF-IDF is a statistical measure that assesses the relevance of a word to a document in a set of documents. Then we will be building predictive models for Bullying, Fake News, and Spamming. For bullying, we will be using Decision Tree, Linear Regression, and Random Forest. For Spamming, we will be using Support Vector Machine, Naïve Bayes, Decision Tree, Random Forest, and K - Nearest Neighbor. Then for fake news, we will be using Random Forest, Linear Regression, and Decision Tree. At last, based on the outcomes we will predict the probability of occurrence of an event by evaluating Metrics – Accuracy. We conclude that the model with good accuracy is chosen as the best one.

DATASETS

Kaggle is also used to collect real-time unstructured data that has been converted to a structured format. For each (i.e., bullying and non-bullying, fake news, and spamming), unique datasets are collected. The most important task of any machine learning project is data collection. Because the data we feed the algorithms is the input. As a result, the algorithms' efficiency and accuracy are determined by the accuracy and quality of the data collected.

PREPROCESSING

Data Preprocessing is the process of converting raw data into a usable and efficient format by removing noise and inconsistencies such as punctuation, date, title, Stop Words, Stemming, Lowercase conversion, and, if necessary, lemmatization. As a result, it generates consistent and reliable data, which improves the efficiency of the training data for analysis and allows for accurate decision-making.

	title	text	subject	date	target
0	Croatia wants to adopt euro within 7-8 years: ...	ZAGREB (Reuters) - Croatia aims to become a eu...	worldnews	October 30, 2017	true
1	House narrowly passes measure paving way for T...	WASHINGTON (Reuters) - The U.S. House of Repre...	politicsNews	October 26, 2017	true
2	Oklahoma rejects Russian request to monitor el...	(Reuters) - Oklahoma voting officials have den...	politicsNews	October 21, 2016	true

1. RemovingDate(attributenotusedforanalysis)

2. Removingthetitle(wewillonlyusethe text)

	title	text	subject	target
0	Croatia wants to adopt euro within 7-8 years: ...	ZAGREB (Reuters) - Croatia aims to become a eu...	worldnews	true
1	House narrowly passes measure paving way for T...	WASHINGTON (Reuters) - The U.S. House of Repre...	politicsNews	true
2	Oklahoma rejects Russian request to monitor el...	(Reuters) - Oklahoma voting officials have den...	politicsNews	true

	text	subject	target
0	ZAGREB (Reuters) - Croatia aims to become a eu...	worldnews	true
1	WASHINGTON (Reuters) - The U.S. House of Repre...	politicsNews	true
2	(Reuters) - Oklahoma voting officials have den...	politicsNews	true

3. Convert to Lowercase

	text	subject	target
0	zagreb (reuters) - croatia aims to become a eu...	worldnews	true
1	washington (reuters) - the u.s. house of repre...	politicsNews	true
2	(reuters) - oklahoma voting officials have den...	politicsNews	true

4. Remove Punctuation

	text	subject	target
0	zagreb reuters croatia aims to become a euro ...	worldnews	true
1	washington reuters the us house of representa...	politicsNews	true
2	reuters oklahoma voting officials have denied...	politicsNews	true

5. Remove Stop Words

	text	subject	target
0	zagreb reuters croatia aims become euro zone m...	worldnews	true
1	washington reuters us house representatives he...	politicsNews	true
2	reuters oklahoma voting officials denied reque...	politicsNews	true
3	world reeling today britain made shocking deci...	News	fake

6. Stemming
Stemming is the process of removing or stemming the last few characters of a word, which frequently results in incorrect meanings and spelling.

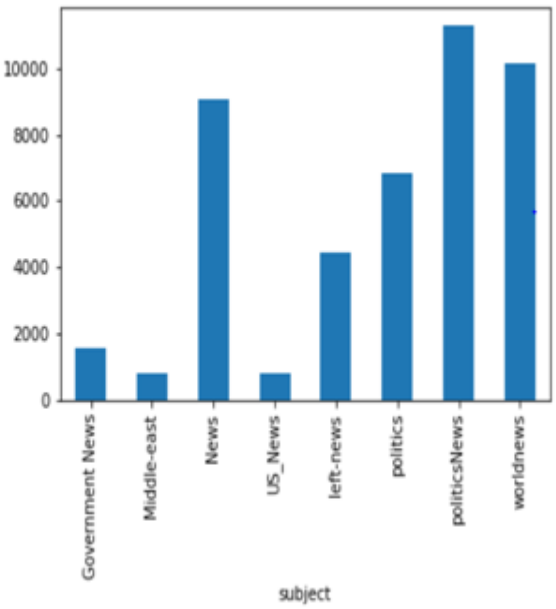
e.g., adjustable - > adjust
formality -> formal

7. Lemmatisation
Lemmatization takes the context into account when converting a word to its meaning base form.

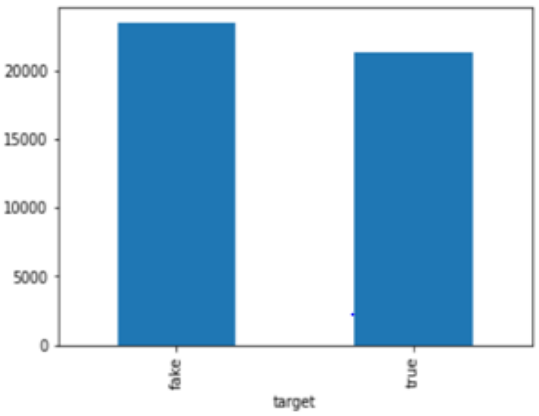
e.g., playing, plays, played - > play
am, are, is -> be

DATAEXPLORATION

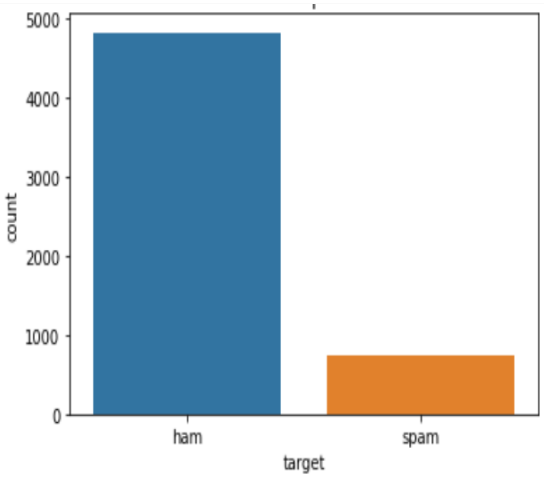
We use visual exploration to figure out what's in a dataset and what the data's attributes are Size, amount of data, completeness of data, correctness of data, and possible relationships are the characteristics here.



Articles per subject



No of Fake and Real Articles



Distribution of Spam and Ham

FEATURE EXTRACTION

Before fitting the train data and test data into a model. We perform feature extraction to vectorize the data.

1. BAG OF WORDS

It's a way of representing text data when using machine learning algorithms to model text. It describes the order in which words appear in a document. The model only cares about whether or not known words appear in the document, not where they appear.It entails two steps:
i) A lexicon of well-known terms.
ii) A metric for determining the presence of well-known words.

Step 1: Collect Data

It was the best of times.
It was the worst of times.
It was the age of wisdom.

Step 2: Design the Vocabulary

We can now create a list of every word in our model vocabulary. The following are the unique words (ignoring case and punctuation):

“it”, “was”, “the”, “best”, “of”, “times”, “worst”, “age”, “wisdom”

From a corpus of 18 words, that's a vocabulary of 9 words.

Step 3: Create Document Vectors

Create a binary vector from the first document ("It was the best of times").The simplest scoring method is to assign a Boolean value to the presence of words, with 0 indicating absence and 1 indicating presence.The document's scoring would be as follows:

“it” = 1 “was” = 1 “the” = 1 “best” = 1“of” = 1 “times” = 1 “worst” = 0 “age” = 0“wisdom” = 0

"it was the best of times" = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

2. TF – IDF

The TF-IDF is a statistical measure that assesses the relevance of a word to a document in a set of documents. It incorporates two ideas:
i)Term Frequency
ii)Inverse Document Frequency

Text 1	i love natural language processing but i hate python
Text 2	i like image processing
Text 3	i like signal processing and image processing

Step 1: Create a term frequency matrix with documents as rows and distinct terms as columns across all documents. Count the number of times each word appears in each text.

Term frequency = No of Repeatation of words in sentence / No of Words in a sentence

	and	but	hate	i	image	language	like	love	natural	processing	python	signal
Text 1	0	1	1	2	0	1	0	1	1	1	1	0
Text 2	0	0	0	1	1	0	1	0	0	1	0	0
Text 3	1	0	0	1	1	0	1	0	0	2	0	1

Step 2: Compute inverse document frequency (IDF) using the explained formula

IDF = log (No of sentences / No of sentences containing words)

Term	and	but	hate	i	image	language	like	love	natural	processing	python	signal
IDF	0.47712	0.47712	0.4771	0	0.1760913	0.477121	0.1760913	0.477121	0.47712125	0	0.477121	0.477121

Step 3: Multiply TF matrix with IDF respectively

	and	but	hate	i	image	language	like	love	natural	processing	python	signal
Text 1	0	0.47712	0.4771	0	0	0.477121	0	0.477121	0.47712125	0	0.477121	0
Text 2	0	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0
Text 3	0.47712	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0.477121

MODELLING

We divide the data into train and test groups when modeling. This is used to estimate the performance of machine learning algorithms, which are algorithms that are used to make predictions using data that was not used to train the model. The model is fitted with the training data, and the test data is used to predict. It's a quick and painless method for comparing the performance of machine learning algorithms for predictive-based modeling problems.

The following are some of the supervised machine learning algorithms that were used in this study:

Spamming	Fake News	Bullying/Non-Bullying
Naïve Bayes	Decision Tree	Naïve Bayes
SVM	Logistic Regression	Logistic Regression
KNN	Random Forest	SVM
Decision Tree		Decision Tree
Random Forest		

5. PERFORMANCE ANALYSIS

The following are the predicted evaluation metrics for Fake News, Cyber Bullying and spamming:

FAKE NEWS	
Model	Accuracy
Decision Tree	99.54%
Logistic Regression	98.74%
Random Forest	99.12%

BULLYING & NON BULLING	
Model	Accuracy
Decision Tree	85.82%
Naïve Bayes	76.58%
SVM	90.96%
Random Forest	95.24%

SPAMMING	
Model	Accuracy
Naïve Bayes	98.26%
SVM	83.43%
KNN	86.06%
Decision Tree	97.30%
Random Forest	97.96%

6. CONCLUSION

False information and bullying on social media have become serious issues in recent years, and a system that can detect such texts would undoubtedly be beneficial. As a result, text

analysis is performed to identify inconsistencies in the text, which is then subjected to various supervised machine learning algorithms to determine the most precise estimation to resolve the issues. This way, we can assist people in making more informed decisions, as well as ensuring that they are not duped into believing what others want them to believe. Bullying, fake news, and spamming can all be mitigated as a result of this.

FUTUREENCHANTMENTS

Testing and training corpora can be tailored to a specific domain in the future, as corpus vocabulary varies by domain.

REFERENCES

[1] M. Dadvar, R.B. Trieschnigg and F.M.G. de Jong. "Experts and Machines against Bullies: A Hybrid Approach to Detect Cyber bullies". In 27th Canadian Conference on Artificial Intelligence, University of Waterloo, Montral, Canada, 2014.

[2] A. Kontostathis, K. Reynolds, A. Garron, and L. Edwards. 2013. "Detecting cyber bullying: query terms and techniques". In Proceedings of the 5th WebSci 2013. ACM, New York.

[3] M. Dadvar and F. de Jong. 2012. "Cyber bullying detection: a step toward a safer internet yard". In Proceedings of the 21st International Conference on World Wide Web (WWW '12 Companion). ACM, New York, NY, USA, 121-126.

[4] K. Dinakar, R. Reichart, and H. Lieberman, "Modeling the Detection of Textual Cyber bullying," MIT. International Conference on Weblog and Social Media. Barcelona, Spain, 2011.

[5] B. Erçahin, Ö. Aktaş, D. Kilinç, and C. Akyol, ''Twitter fake account detection,'' in Proc. Int. Conf. Comput. Sci. Eng. (UBMK), Oct. 2017, pp. 388–392.

[6] T. Wu, S. Wen, Y. Xiang, and W. Zhou, ''Twitter spam detection: Survey of new approaches and comparative study,'' Comput. Secur., vol. 76, pp. 265–284, Jul. 2018.

[7] N. Eshraqi, M. Jalali, and M. H. Moattar, ''Detecting spam tweets in Twitter using a data stream clustering algorithm,'' in Proc. Int. Congr. Technol., Commun. Knowl. (ICTCK), Nov. 2015, pp. 347–351.

[8] C. Buntain and J. Golbeck, ''Automatically identifying fake news in popular Twitter threads,'' in Proc. IEEE Int. Conf. Smart Cloud (SmartCloud), Nov. 2017, pp. 208–215.

[9] C. Chen, J. Zhang, Y. Xie, Y. Xiang, W. Zhou, M. M. Hassan, A. AlElaiwi, and M. Alrubaian, ''A performance evaluation of machine learning-based streaming spam tweets detection,'' IEEE Trans. Comput. Social Syst., vol. 2, no. 3, pp. 65–76, Sep. 2015.

REFERENCE

1. A. Gupta and R. Kaushal, ‘‘Improving spam detection in online social networks,’’ in Proc. Int. Conf. Cogn. Comput. Inf. Process. (CCIP), Mar. 2015, pp. 1–6.
2. A. Gupta, H. Lamba, and P. Kumaraguru, ‘‘1.00 per RT #BostonMarathon # prayforboston: Analyzing fake content on Twitter,’’ in Proc. eCrime Researchers Summit (eCRS), 2013, pp. 1–12.
3. B. Erçahin, Ö. Aktaş, D. Kilinç, and C. Akyol, ‘‘Twitter fake account detection,’’ in Proc. Int. Conf. Comput. Sci. Eng. (UBMK), Oct. 2017, pp. 388–392.
4. C. Buntain and J. Golbeck, ‘‘Automatically identifying fake news in popular Twitter threads,’’ in Proc. IEEE Int. Conf. Smart Cloud (SmartCloud), Nov. 2017, pp. 208–215.
5. C. Chen, J. Zhang, Y. Xie, Y. Xiang, W. Zhou, M. M. Hassan, A. AlElaiwi, and M. Alrubaian, ‘‘A performance evaluation of machine learning-based streaming spam tweets detection,’’ IEEE Trans. Comput. Social Syst., vol. 2, no. 3, pp. 65–76, Sep. 2015.
6. C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min, ‘‘Statistical features-based real-time detection of drifted Twitter spam,’’ IEEE Trans. Inf. Forensics Security, vol. 12, no. 4, pp. 914–925, Apr. 2017.
7. F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, ‘‘Detecting spammers on Twitter,’’ in Proc. Collaboration, Electron. Messaging, AntiAbuse Spam Conf. (CEAS), vol. 6, Jul. 2010, p. 12.

8. F. Concone, A. De Paola, G. Lo Re, and M. Morana, ‘‘Twitter analysis for real-time malware discovery,’’ in Proc. AEIT Int. Annu. Conf., Sep. 2017, pp. 1–6.
9. F. Fathaliani and M. Bouguessa, ‘‘A model-based approach for identifying spammers in social networks,’’ in Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA), Oct. 2015, pp. 1–9.
10. G. Stafford and L. L. Yu, ‘‘An evaluation of the effect of spam on Twitter trending topics,’’ in Proc. Int. Conf. Social Comput., Sep. 2013, pp. 373–378.
11. M. Mateen, M. A. Iqbal, M. Aleem, and M. A. Islam, ‘‘A hybrid approach for spam detection for Twitter,’’ in Proc. 14th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST), Jan. 2017, pp. 466–471.
12. N. Eshraqi, M. Jalali, and M. H. Moattar, ‘‘Detecting spam tweets in Twitter using a data stream clustering algorithm,’’ in Proc. Int. Congr. Technol., Commun. Knowl. (ICTCK), Nov. 2015, pp. 347–351.
13. S. Gharge, and M. Chavan, ‘‘An integrated approach for malicious tweets detection using NLP,’’ in Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT), Mar. 2017, pp. 435–438.
14. S. J. Soman, ‘‘A survey on behaviors exhibited by spammers in popular social media networks,’’ in Proc. Int. Conf. Circuit, Power Comput. Technol. (ICCPCT), Mar. 2016, pp. 1–6.
15. T. Wu, S. Wen, Y. Xiang, and W. Zhou, ‘‘Twitter spam detection: Survey of new approaches and comparative study,’’ Comput. Secur., vol. 76, pp. 265–284, Jul. 2018.