FAKE NEWS DETECTION USING MACHINE LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

KALICHETI BHARADWAJ REDDY (211416104127)

V.JITHESH BABU (211416104118)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY: CHENNAI 600 025

SEPTEMBER 2020

BONAFIDE CERTIFICATE

Certified that this project report **"FAKE NEWS DETECTION USING MACHINE LEARNING TECHNIQUES"** is the bonafide work of **KALICHETI BHARADWAJ REDDY (211416104127), V.JITHESH BABU (211416104118),** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.S. MURAGAVALLI, M. E, Ph.D.	**Mr. A. KARTHIKEYAN, M. Tech.**
HEAD OF THE DEPARTMENT	**SUPERVISIOR**
	ASSISTANT PROFESSOR
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
Panimalar Engineering College,	Panimalar Engineering College,
Poonamallee, Chennai 600123	Poonamallee, Chennai 600123

Certified that the above candidate is examined in the Anna University Project Viva-Voice Examination held on _________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

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

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C. VIJAYARAJESWARI, Thiru.C. SAKTHIKUMAR, M.E.,** 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 and encouragement provided to us throughout the course.

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

We would like to thank my **Project Guide Mr.A.KARTIKEYAN, Asst.Professor** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

KALICHETI BHARADWAJ REDDY

V.JITHESH BABU

ABSTRACT

News is one of the most important aspect in knowing and sharing information in society. Publishing news through social media or Internet news articles has taken over the traditional methods of how news used to publish. But due to anonymity and lack of verification of the information posted in the news articles lot of misleading articles (Fake news) is being published on the internet. To tackle the problem researchers have been developing various systems to detect the fake news but most systems are detecting the fake news using word count or sentimental analysis of the news articles but there are various problems arises with both the systems, by using word count it does not understand the meaning of the article so the authors can easily hoax the system while coming to sentimental analysis news articles can be fake whether they are publishing positive or negative news and there are systems where it uses the Naïve Bayes to prove it is the most efficient, so we are introducing the Xgboost and Passive-Aggressive classifier to prove they are most suitable for building a model for fake news and to find the pattern of the fake news article and differentiate from the real news.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATION

Abbreviation	**Meaning**
ML	Machine Learning
CM	Confusion Matrix
NB	Naïve Bayes
PAC	Passive Aggressive Classifier
XGB	Xgboost
Tf-idf	Term frequency inverse document frequency

LIST OF FIGURES

CHAPTER 1

1.1 Domain Overview

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

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they "learn" about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modelling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be biclass (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problem are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

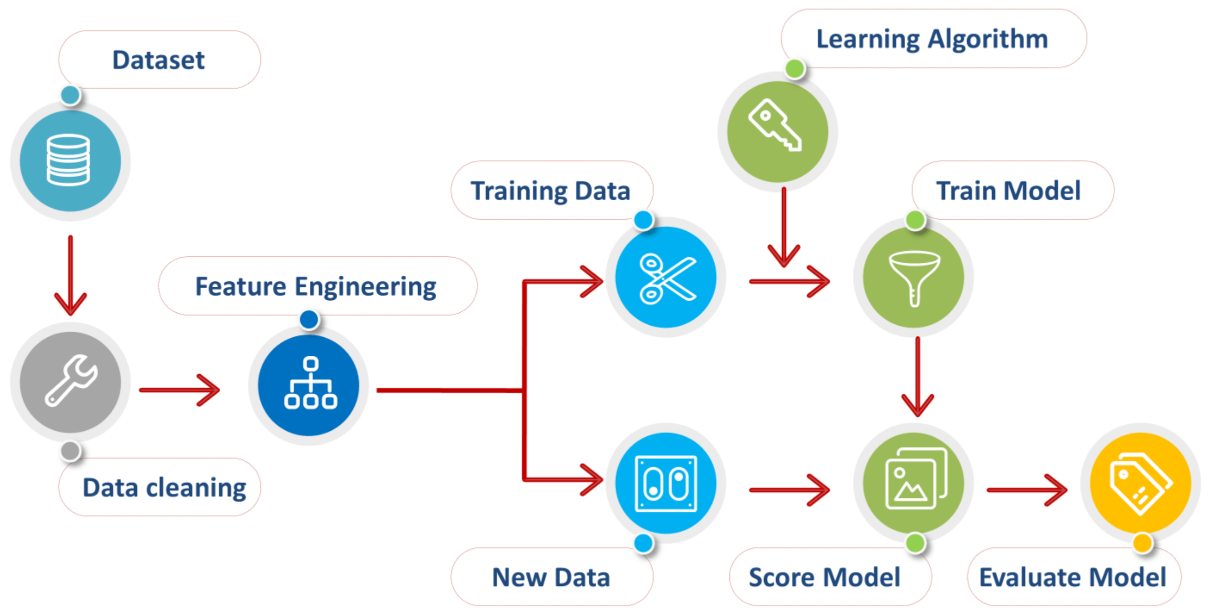


Fig 1.1 Process of Machine Learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is y = f(X). The goal is to approximate the mapping function so well that when the new input data (X) can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is the output variable is a category such as “red” or “blue”.

1.2 FAKE NEWS:

Fake news is one of the major problems in the modern world but the issue of fake news is highlighted during the 2016 US presidential election than ever before. A lot of media outlets have published fake stories about both the leading candidates of democratic and republican parties. Out of the fake stories that have published favoring Donald Trump they are shared over 30 million times and in the case of Hilary Clinton 8 million shares have been recorded [6]. Which played a major part in the election, you can say the fake news have turned the result of the election. Not only in the US there are many countries where fake news has played a major part in their election campaign one of the most prominent examples is the 2016 general election of INDIA. WhatsApp has become the major platform for the spread of misinformation or false information in India. A major news publisher in the world BBC has called WhatsApp the ‘black hole’ in the Indian elections. Not only for the political reasons there can be major economic problems arises due to the spread of fake news stock markets that can be affected because of it. So, a counterfeit of fake news is more necessary than ever in the history of news publishing.

Before distinguishing the news as fake or real we have to understand what news can we consider fake and what to consider real and have differentiated both of them. They are two types of fake news that can be published. 1. news explaining the facts 2. Misleading articles. While finding whether the news that explains the facts is fake or real is not quite possible using because they subjective to the author and type of issue. But there is a way to distinguish the misleading articles we can find the pattern followed in the fake and real news stories and differentiate them. The misleading articles is mostly used by the political parties by which they spread false information about opposition parties to gain public affection for them.

The major goal is to find various patterns here the patterns are nothing but the vectors. Since we are using the machine learning methods, we have to find the dataset which consists of accurate fake and real stories creating the correct dataset is essential for the efficiency of the model. We are going find using the title and summary of the news article using the NLP and newspaper techniques.

1.3 Preparing the dataset

Preparing the dataset is the difficult part in the process we have to make sure to create a dataset such that it contains certain news articles so we can find the pattern involved in writing the fake or real articles. Finding articles for fake news is easy because Facebook conducted a challenge in Kaggle to create a dataset for fake news articles approximately 12,000 Articles have been gathered. But finding the real dataset is a harder process so we scrapping news articles from web-only from very trusted sources such as Guardian, New York Times, BBC. In addition to these, there is a data repository known as Fake news net with news content, social content and dynamic information we are going to randomly select data from these and create a new fake or real dataset for predicting of fake news articles.

The dataset we are going to create will contain articles only about the political news articles So that we can clearly distinguish the data set and process the information to the model.

1.) Gathering fake news dataset

We have downloaded political real and fake news dataset from https://github.com/KaiDMML/FakeNewsNet/tree/master/dataset
And combined them with the Kaggle fake news dataset to obtain the dataset for fake news.

2.) Gathering real news dataset

We have scrapped the net by downloading the news articles from various high-profile news sites such as Guardian, New York Times, Bloomberg etc to get the real news dataset.

3.) Combining both and eliminating garbage data

At last we combined the both datasets and eliminated the empty labels and created a new political dataset for fake news detection.

DATASET MODEL:

ID NO	**TITLE**	**TEXT**	**FAKE/REAL**

Table 1: Dataset Model Table

1.4 Existing Model

In previous model fake news system devised based on the combination of sentimental analysis and naïve Bayes algorithm the main concept of the model is to predict the news is fake or real based on the sentiment generated by the article by users. It takes into consideration that if the sentiment of the article is negative it is fake news or else real news. It does not blindly follow the process it combines with naïve Bayes and train the machine to produce the results.

Steps:

1. Uses a dataset

2.Use twitter data as training dataset to predict the sentiment of the article which is used.

3.Merges the cosine similarity + tfidf similarity + sentiment

4. Train the model using naïve Bayes and random forest

5. Make prediction and analyse the results

It is able to obtain an accuracy rate of average 0.80 which is pretty good considering the sentiment value.

Drawbacks:

1.) Since it is using sentiment as a factor in the training process there is a major bias in it.

2.) If the sentiment of the news is negative there isn't any possible explanation that it is fake news and vice versa

3.) The recall value for the given system is very less which is a major factor in fake news systems.

1.5 Proposed System:

1.) We are going to use a find the fake or real news about a specific category of news (Political)

2.) Preparing a Dataset is most important in this process the more accurate the dataset is the most efficient results we can get from the system.

3.) Preparing the dataset is the difficult part in the process we have to make sure to create a dataset such that it contains certain news articles so we can find the pattern involved in writing the fake or real articles. Finding articles for fake news is easy because Facebook conducted a challenge in Kaggle to create a dataset for fake news articles approximately 12,000 Articles have been gathered. But finding the real dataset is a harder process so we scrapping news articles from web-only from very trusted sources such as Guardian, New York Times, BBC. In addition to these, there is a data repository known as Fake news net with news content, social content and dynamic information we are going to randomly select data from these and create a new fake or real dataset for predicting of fake news articles.

4.) We are going to use Naïve Bayes, Passive Aggressive Classifier and Xgboost algorithm to train the models with the combination of count and tfidf vectorizer.

5.) As told before recall is the most important when creating fake news model so we are going to prove Xgboost algorithm provides more efficiency than naïve Bayes for fake news model.

CHAPTER 2

LITERATURE SURVEY

2.1 General

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

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them. Loan default trends have been long studied from a socio-economic stand point. Most economics surveys believe in empirical modelling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which it is observing now. Some of the surveys to understand the past and present perspective of loan approval or not.

2.2 Review of Literature Survey

1.) B. Bhutani, N. Rastogi, P. Sehgal and A. Purwar, "Fake News Detection Using Sentiment Analysis," *2019 Twelfth International Conference on Contemporary Computing (IC3)*, Noida, India, 2019, pp. 1-5, doi: 10.1109/IC3.2019.8844880.

Detecting fake news using the content and sentiment of the information provided in the news article if the sentiment of the article is mostly negative is viewed as fake news or else real but we cannot directly follow it positive news can also be considered as fake news. Here they created a hybrid system where it takes sentiment as a factor in the detection of fake news. The en-coded vector is returned with length of the entire vocabulary (bag of words) and an integer count for the number of times each word appeared had in the document. [2]

2.) C. K. Hiramath and G. C. Deshpande, "Fake News Detection Using Deep Learning Techniques," *2019 1st International Conference on Advances in Information Technology (ICAIT)*, Chikmagalur, India, 2019, pp. 411-415, doi: 10.1109/ICAIT47043.2019.8987258.

But not only using the hybrid techniques there are direct approaches to solve the fake news problem here the author used deep learning techniques to shows that deep learning algorithms will work better in terms of time and accuracy than the regular text classification algorithms. The DNN techniques work better than all the techniques but it needs more space than all the algorithms used in the process.[3]

3.) R. K. Kaliyar, "Fake News Detection Using A Deep Neural Network," 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018, pp. 1-7, doi: 10.1109/CCAA.2018.8777343.

By using the deep neural network the author classified the fake news but he cannot use the traditional neural network because they cannot key track of what happened in the previous cycles of the process, to overcome this problem the author uses LSTM combined with CNN to tackle the problem and creates the building blocks of RNN(recurrent neural network). It works with the combination of VDCNN and GRU, the benefit of feature extraction is also possible using this model.[4]

4.) Fake news classifies or detection classifies into various type content-based, social context-based hybrid based in the content-based there is a linguistic cue, visual-based in the social context-based stance based, propagation-based in the hybrid-based we can combine various techniques to obtain the result. Out of all the techniques, the author concluded more specific results can be obtained using the hybrid model. [5]

5.) Conroy, N. J., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. Proceedings of the Association for Information Science and Technology

Deception detection techniques have utilized to

Find out about the fake news from a long time here the author uses the linguistic techniques and network approaches. In linguistic techniques, the content is extracted and analyzed to find the pattern of the deceptive messages the main technique followed in the process is by using the n-gram model but only using the n-gram model is also its big disadvantage so the author prepared a hybrid model with the combination of network approaches in which data such as message metadata or structured knowledge network queries can be harnessed to provide aggregate deception measures. Both forms typically incorporate machine learning techniques for training [1].

6.) Fake News Net:

A Data Repository with News Content, Social Context and Dynamic Information for Studying Fake News on Social Media Social media has become a popular means for people to consume news. Meanwhile, it also enables the wide dissemination of fake news, i.e., news with intentionally false information, which brings significant negative effects to the society. Thus, fake news detection is attracting increasing attention. However, fake news detection is a non-trivial task, which requires multi- source information such as news content, social context, and dynamic information. First, fake news is written to fool people, which make it difficult to detect fake news simply based on news contents. In addition to news contents, we need to explore social contexts such as user engagements and social behaviors. For example, a credible user's comment that "this is fake news" is a strong signal for detecting fake news. Second, dynamic information such as how fake news and true news propagate and how users' opinions toward news pieces are very important for extracting useful patterns for (early) fake news detection and intervention. Thus, comprehensive datasets which contain news content, social context, and dynamic information could facilitate fake news propagation, detection, and mitigation; while to the best of our knowledge, existing datasets only contains one or two aspects. Therefore, in this paper, to facilitate fake news related researches, we provide a fake news data repository Fake News Net, which contains two comprehensive datasets that includes news content, social context, and dynamic information. We present a comprehensive description of datasets collection, demonstrate an exploratory analysis of this data repository from different perspectives, and discuss the benefits of Fake News Net for potential applications on fake news study on social media.

CHAPTER 3

REQUIREMENTS AND TECHNOLOGIES

3.1 FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine that it would be financially and technically feasible to develop the product.

3.1.1 TECHNICAL FEASIBILITY

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

3.1.2 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. The enhancement of the existing system doesn’t incur any kind of drastic increase in the expenses. Python is open source and readily available for all users. Since the project is running in python and Anaconda notebook hence is cost efficient.

3.2 HARWARE REQUIREMENTS

- Processor : Intel Pentium Dual Core 2.00GHz
- Hard disk : 40 GB
- RAM : 2 GB (minimum)

3.3 SOFTWARE REQUIREMENTS

- Anaconda navigator
- Python 3.6.4 Version

3.4 SUPERVISED LEARNING

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labelled by humans, and unsupervised learning which provides the algorithm with no labelled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output Y = f(X). The goal is to approximate the mapping function so well that when you have new input data. That you can predict the output variables (Y) for that data. Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Trees and support vector machines. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics. Supervised learning problems can be further grouped into Regression and Classification problems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification.

3.5 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:

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 our observation belongs 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. We train a model, called Classifier on this data set, and use that model to predict whether a certain patient will have the

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

CHAPTER 4

ARCHITECTURE AND MODULES

4.1 SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal ascription and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

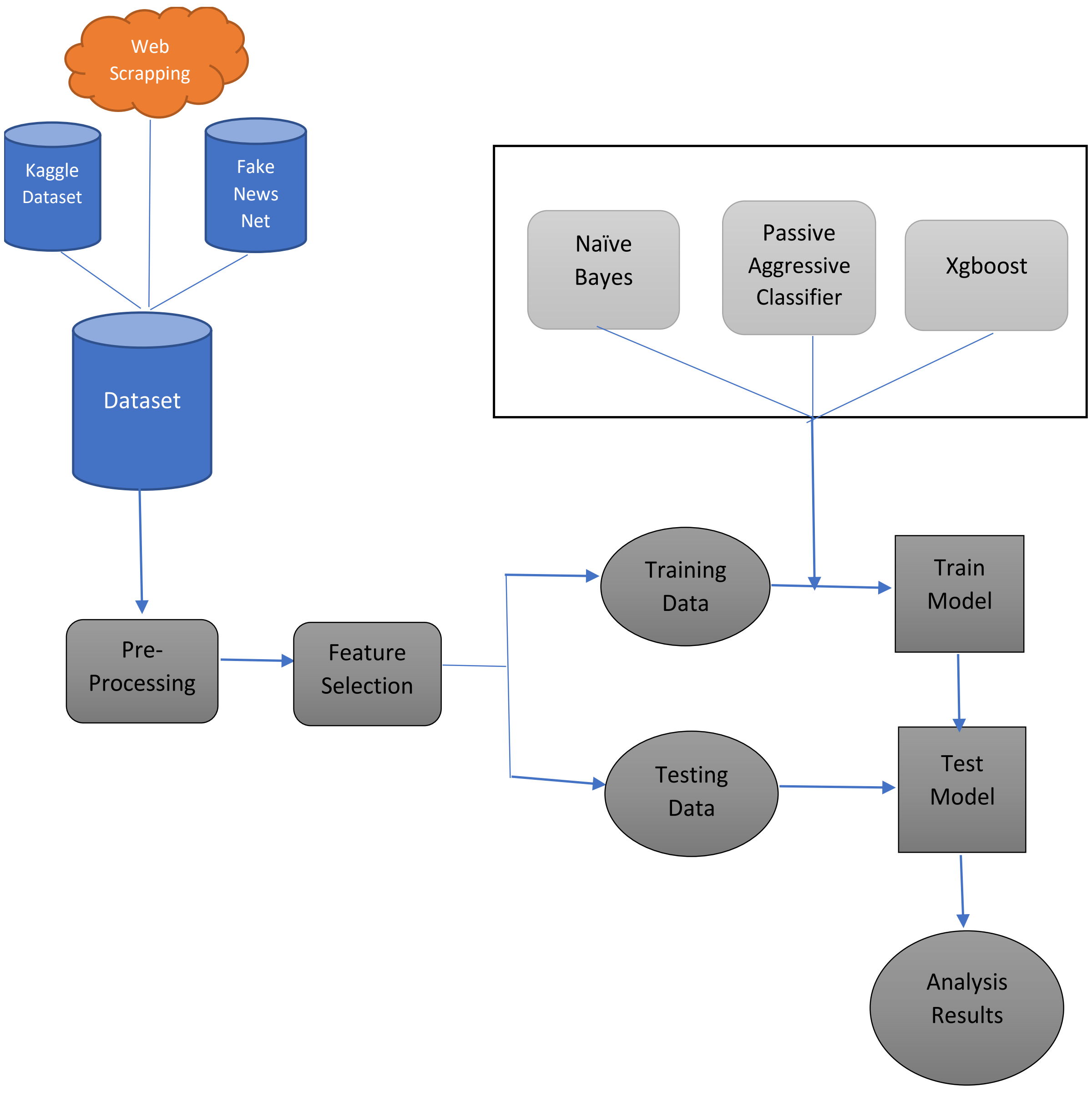


Fig 1.2 System Architecture

STEPS:

1. For effective prediction of the type of news ideal dataset is necessary so we have created a dataset by gathering the fake news articles from the Facebook competition conducted in the Kaggle, while coming to the real news dataset we have scrapped the major newspapers such as Guardian ,New York Times, Bloomberg etc. In addition, we have gathered dataset from Fake news net an online data repository whose main goal to gather news and separate them fake and real, we have eliminated all the news articles except political news and created a dataset for political news exclusively.

2. To process the dataset it was taken as input to the model by which we have predicted the news articles were fake or real

3. In the pre-processing phase of the process we usually perform these functions data cleaning, Data integration, Data transformation, Data reduction, data discreditation each of the sub step has its importance and works accordingly to the process. While gathering large number of datasets there is a possibility some of the data in some classes may not be available so pre-processing the data is useful to move forward in the process.

4. By using Feature Scaling to the dataset we had make sure all the data is of small or in near values in terms of Unicode or vector to make sure that calculation part goes easier and the algorithms will not take more time due to the process.

5. The basic idea of machine learning is to train the machine with data of the certain problem and test it to obtain the results this is one of the important part of the process normally 70 percent of the data will goes to the training part and the remaining 30 percent will go to the testing part .we have predicted the data allocated to the testing part.

6. In this step we have introduced the count vectorizer and Tfidf vectorizer each process the words in separate manner in the count vectorizer it processes the data by number of times it appears in the respective system it can be biased based on the data. In the Tfidf we are going to use the frequency of the word in the data which is very important to find the pattern in fake news each train and test tis label to them so that we can use them in various algorithms in the next step.

7. We have use various algorithms Multinomial naïve Bayes, Passive aggressive classifier, Xgboost algorithms in combination with the count and tfidf to process the data.

i.)In Multinomial naïve Bayes are mostly used in NLP problems tracks the tag of text and returns the tag with highest probability to the given input but before performing the operations we have to change the text into numbers so that we can perform the calculations there nlp techniques such as stemming , removing stop words , feature Engineering to process it but in most of the cases we may not find the newly added word in the training dataset so we perform the smoothing process to find out the probability the most common technique in terms of smoothing is Laplace smoothing

ii.) Passive aggressive classifier is a combination of Classification, Regression and Uni class it follows the following process 1. Receive instance 2. Predict target value 3. Receive true target if it suffers loss it updates the hypothesis, it is the reason it called as aggressive the goal in passive aggressive classifier is to minimize the loss by combining and updating the classification, Regression and Uniclass methods such that end result is a common algorithmic framework and an accompanying analysis. In short, we can say that passive regressive algorithms are family of algorithms for large scale learning.

iii) Xgboost is an optimized gradient boosting algorithm through parallel processing, tree -priming handling missing values and regularization to avoid overfitting /bias it has cache awareness and out of core computing ,efficient in handling of missing data, In built cross validation capability it does not require feature scaling ,pre-processing the training time is less when compared to other algorithms.

4.2 MODULES

4.2.1 Pre-Processing the Data

After Creating the dataset, we have to import the necessary libraries for the correct execution of model then we have to divide the columns in the dataset one side consists of the id, text and title, the other side consists of the label whether it is fake or real news. The algorithms cannot process the text in its original form so we are going to transform every text in Unicode before that we have to eliminate any N/A means vacant data from the dataset and Split them into training and test set.

4.2.1.1 Creating the dataset

1. For the effective prediction of the type of news ideal dataset is necessary so we have created a dataset by gathering the fake news articles from the Facebook competition conducted in the Kaggle, while coming to the real news dataset we have scrapped the major newspapers such as Guardian, New York Times, Bloomberg, etc. Besides, we have gathered dataset from Fake news net an online data repository whose main goal to gather news and separate them fake and real, we have eliminated all the news articles except political news and created a dataset for political news exclusively.

2. To process the dataset it was taken as input to the model by which we have predicted the news articles were fake or real

4.2.1.2 Pre-processing

3. In the preprocessing phase of the process, we usually perform these functions data cleaning, Data integration, Data transformation, Data reduction, data discreditation each of the sub-step has its importance, and works accordingly to the process. While gathering a large number of datasets there is a possibility some of the data in some classes may not be available so preprocessing the data is useful to move forward in the process.

4. By using Feature Scaling to the dataset we had to make sure all the data is of small or in near values in terms of Unicode or vector to make sure that calculation part goes easier and the algorithms will not take more time due to the process.

4.2.1.3 Splitting the Data

5. The basic idea of machine learning is to train the machine with data of the certain problem and test it to obtain the results this is one of the important parts of the process normally 70 percent of the data will go to the training part and the remaining 30 percent will go to the testing part .we have predicted the data allocated to the testing part

4.2.2 Implementing Algorithms

4.2.2.1Count Vectorizer

Count Vectorizer converts a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts using scipy.sparse.csr_matrix. The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

Create an instance of the CountVectorizer class. If a string, it is passed to _check_stop_list and the appropriate stop list is returned is currently the only supported string value.

If a list, that list is assumed to contain stop words, all of which will be removed from the resulting tokens.

The result bag of words or vocabulary matrix is given as train and test input.

4.2.2.2Tfidf Vectorizer

One issue with simple counts is that some words like “the” will appear many times and their large counts will not be very meaningful in the encoded vectors. Thus, an alternative is to calculate word frequencies, and by far the most popular method is called TF-IDF (Term Frequency – Inverse Document Frequency) which are the components of the resulting scores assigned to each word. Term Frequency: This summarizes how often a given word appears within a document. Inverse Document Frequency: This downscales words that appear a lot across documents.

The Tfidf Vectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents.

Here we used to stop words='English', max_df=0.7, which removes the stops words from dataset text and removes the words which appears in more than 70% of documents.

4.2.2.3 Naïve Bayes Algorithm

Naive Bayes is a probabilistic algorithm that’s typically used for classification problems. Naive Bayes is simple, intuitive, and yet performs surprisingly well in many cases.

There are various types of naïve Bayes algorithms.

Multinomial Naive Bayes:

It estimates the conditional probability of a particular word given a class as the relative frequency of term t in documents belonging to class(c). The variation takes into account the number of occurrences of term t in training documents from class (c), including multiple occurrences.

Boolean Multinomial Naive Bayes:

It works same like Multinomial naive Bayes only change instead of measuring all occurrence in term (t) in document it measures the occurrence only once.

Bernoulli Naive Bayes Model:

It generates Boolean value/indicator about each term of the vocabulary equal to 1 if the term belongs to examining document, if not it marks 0. Non-occurring terms in document are takes into document and they are factored when computing the conditional probabilities and thus the absence of terms is taken into account.
In some cases, we get probability zero so there is a solution known as Laplace Smoothing to counter this issue.

Laplace Smoothing:

It is a technique for smoothing categorical data. A small-sample correction, or pseudo-count, will be incorporated in every probability estimate. Consequently, no probability will be zero. this is a way of regularizing Naive Bayes, and when the pseudo-count is zero, it is called Laplace smoothing
In this problem we are using multinomial naïve Bayes since there is so much data in dataset.

4.2.2.4 Passive Aggressive Classifier:

The passive-aggressive algorithms are a family of algorithms for large-scale learning. They are similar to the Perceptron in that they do not require a learning rate. However, contrary to the Perceptron, they include a regularization parameter C.

For simplicity, let me start with the case of classification.
On each round we receive an instance x*t* and extend a prediction using our current hypothesis w*t*. We then receive the true target y*t* and suffer an instantaneous loss based on the discrepancy between y*t* and our prediction.
Our goal is to make the cumulative loss that we suffer small.
Finally, we update the hypothesis according to the previous hypothesis and the current example.

The setting for regression and uniclass is similar.
In Regression, we predict a real value target w*t* * x*t* and then suffer loss according to the discrepancy between our prediction and the true one.

In Uniclass, the setting is slightly different as we do not have instances. Instead, we extend a centre point w*t*. and hope that y*t* falls within a radius of epsilon from w*t*. We again suffer loss according to the actual distance between y*t* and the centre w*t*.

4.2.2.5 Xgboost Algorithm:

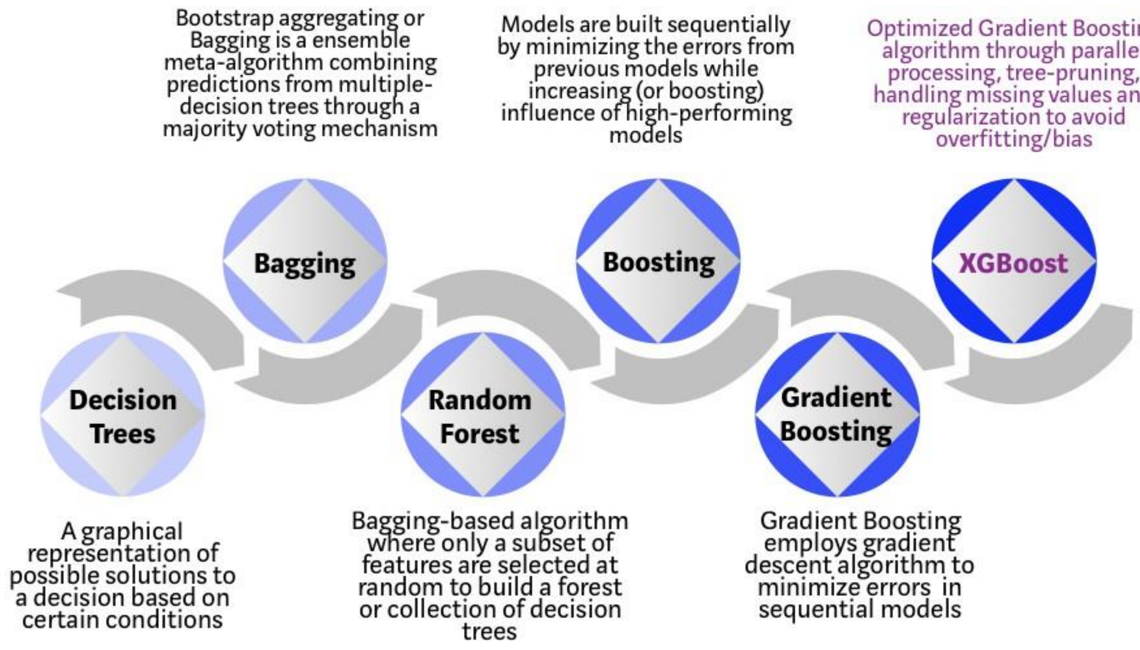

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Fig 1.3 Xgboost explanation

The above image is a representation in which Xgboost algorithm is better than all the algorithms used in the machine learning and what is Xgboost algorithm.

In text Xgboost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) ... A wide range of applications: Can be used to solve regression, classification, ranking, and user-defined prediction problems.

Advantages of Xgboost:

1. Regularization: Xgboost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, Xgboost is also called regularized form of GBM (Gradient Boosting Machine).

2. Parallel Processing: Xgboost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model.

While using Scikit Learn library, **nthread** hyper-parameter is used for parallel processing. **nthread** represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for **nthread** and the algorithm will detect automatically.

3. Handling Missing Values: Xgboost has an in-built capability to handle missing values. When Xgboost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

4. Cross Validation: Xgboost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited value can be tested.

4.3 Comparing Results

4.3.1 Confusion matrix:

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

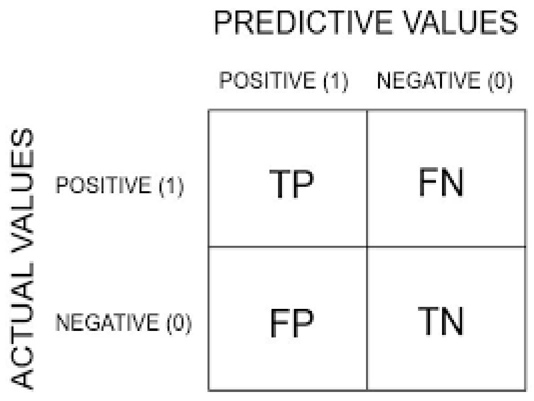


Fig 1.4 Confusion Matrix Explanation

False Positives (FP) A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN) A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP) A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN) A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

Comparing Algorithm with prediction in the form of best recall result:

True Positive Rate (TPR) = TP / (TP + FN) False Positive rate (FPR) = FP / (FP + TN)

Accuracy

The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation.

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. The question that this metric answer is of all passengers that labelled as survived, how many actually survived Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

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

Precision:

The proportion of positive predictions that are actually correct. (When the model predicts default: how often is correct Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labelled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.
Precision = TP / (TP + FP)

Recall

The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict).

Recall = TP / (TP + FN).

There you go! So Recall actually calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

For instance, in fraud detection or sick patient detection. If a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be very bad for the bank. Similarly it is same for fake news detection.

F1 Score: The weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

General Formula F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula F1 Score = 2*(Recall *) / (Recall + Precision1)

4.4 USE CASE DIAGRAM

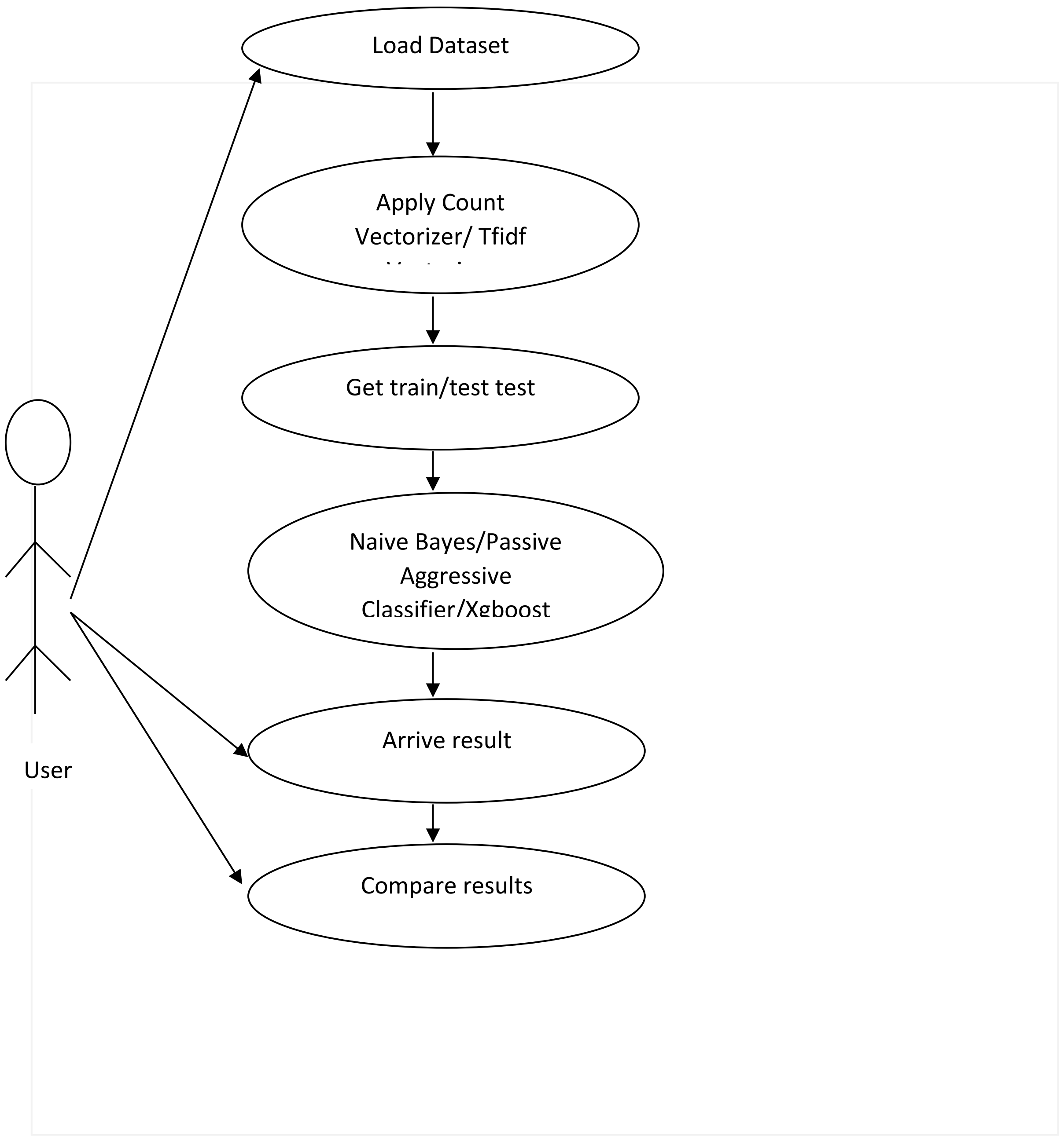

Load Dataset
Apply Count Vectorizer/ Tfidf
Get train/test test
Naive Bayes/Passive Aggressive Classifier/Xgboost
Arrive result
Compare results
User

Fig 1.5 Use Case Diagram

The above figure represents use case diagram, in which user upload dataset The functional modules such as feature extraction and training and prediction are shown. Dataset are analysed for fake or real.

4.5 SEQUENCE DIAGRAM

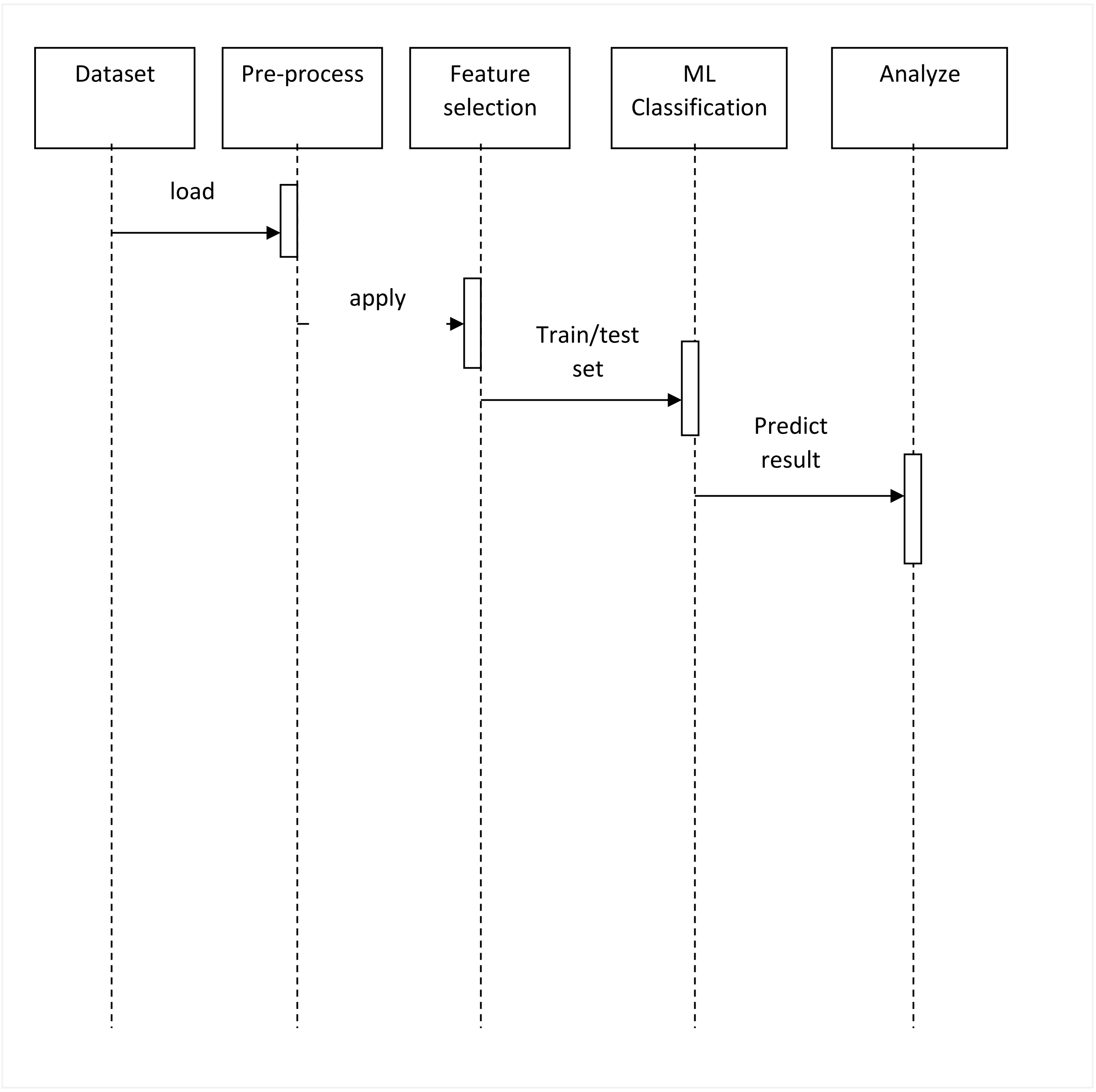


Fig: 1.6 Sequence Diagram

A sequence diagram shows a parallel vertical lines, different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in order in which they occur. The above figure represents sequence diagram, the proposed system's sequence of data flow is represented.

4.6 ACTIVITY DIAGRAM

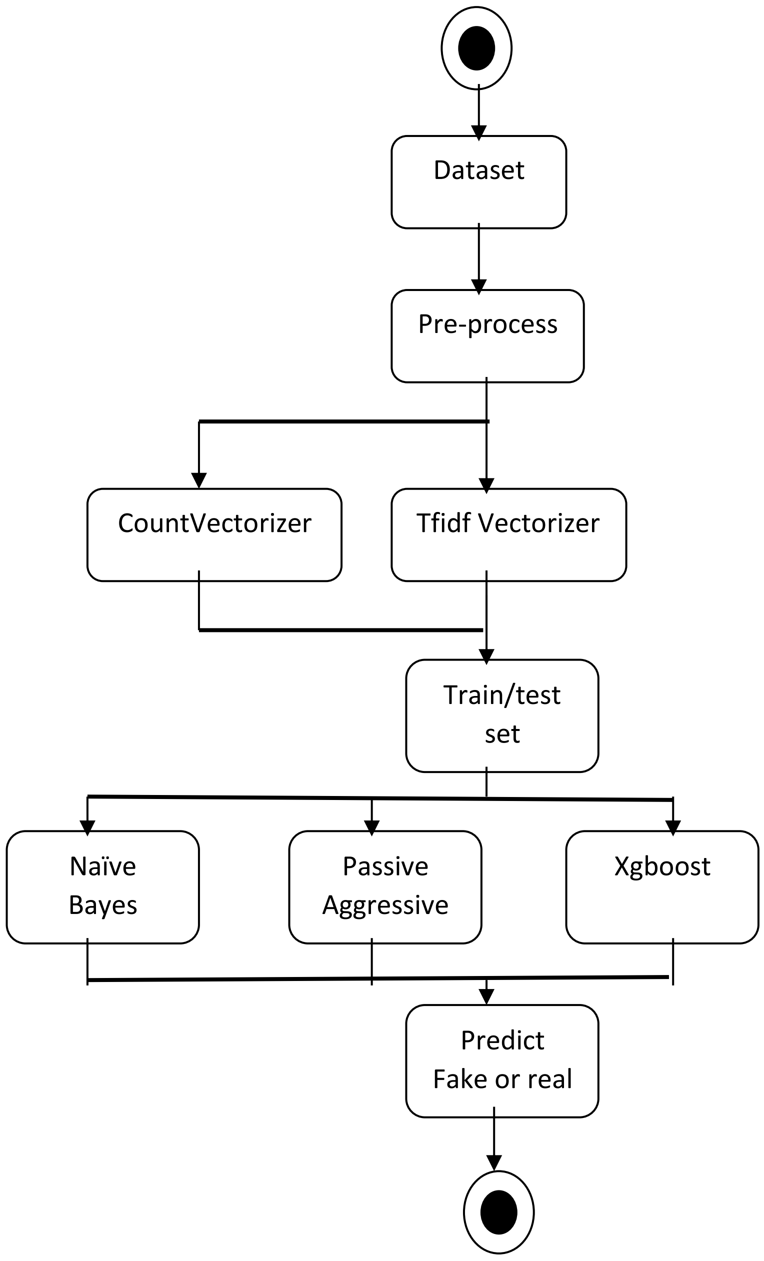


Figure:1.7 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities actions with support for choice, iteration and concurrency. The above figure shows Activity diagrams for fake review detection process.

4.7 COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the

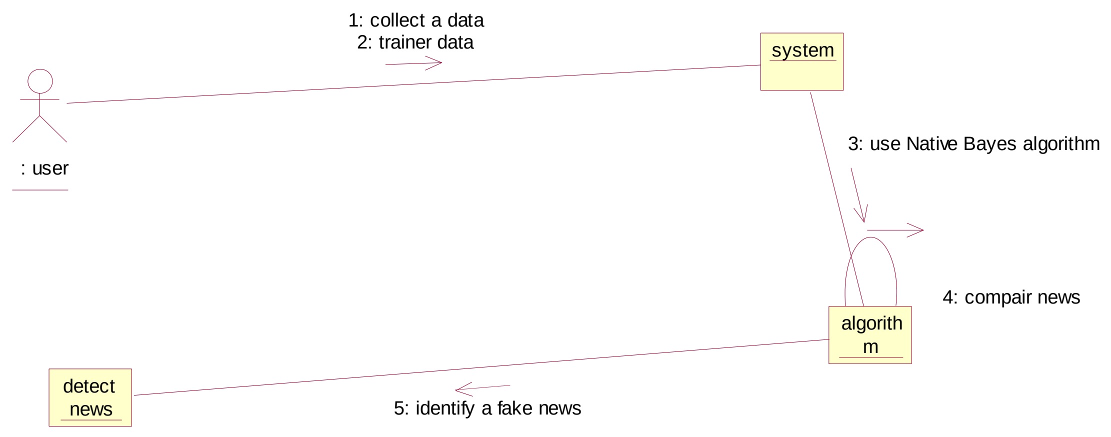


Fig :1.8 Collaboration Diagram

Unified Modelling Language (UML). These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object

IMPLEMENTATION STEPS

- ❖ The proposed application should be able to identify fake or real news. Feature extraction models used are n-count and TF/IDF. We used classification models Naïve Bayes, Passive Aggressive Classifier and Xgboost on the training set to predict the news type.
- ❖ Extract the feature using n-count and TF/IDF
- ❖ Split train and test set
- ❖ Apply machine leaning models Naïve Bayes, Passive Aggressive Classifier and Xgboost on the training set
- ❖ On test set, apply machine learning algorithm Naïve Bayes, Passive Aggressive Classifier and Xgboost.
- ❖ Predict the news type, Compare the machine learning algorithm's accuracy on each feature extraction model

CHAPTER 5

TESTING AND CODING

Testing

5.1 System Testing:

The data set collected for predicting loan customers is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Naïve Bayes, Passive Aggressive Classifier and Xgboost algorithms etc. are applied on the Training set and based on the test result accuracy, Test set prediction is done.

5.2 Software Testing:

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists.

5.3 Unit Testing:

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has

5.4 Test Cases

S.NO	**Test Case ID**	**Test Description**	**Test Procedure**	**Expected Result**	**Actual Result**
1	T101	To check dataset loading	Load collected dataset	Dataset should be loaded to execute	Error to load dataset
2	T102	To check correct dataset format	Load collected dataset	Dataset should be loaded to execute	Check the dataset field and column
3	T103	To check training	Start training dataset	Training should start and system learns data	Alert to user "Dataset is trained"
4	T104	To check prediction	Start prediction by test input	Test should start and output files generated	Alert to user "prediction completed"

Table 1.2 Test cases

5.5 Coding:

```
#-------------------------------------------------------IMPORTING THE LIBRARIES---------------------------------------------------------#
import pandas as pd
from sklearn.model_selection import train_test_split
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.feature_extraction.text import HashingVectorizer
import itertools
import numpy as np
import time

#---------------------------EXTRACTING THE DATASET----------------------------#
kbr = pd.read_csv("fake_or_real_news.csv")
z = kbr.label
kbr.drop("label", axis=1)
 #--------------------DIVING TRAINING AND TEST SETS----------------------------#
Ktrain, Ktest, Btrain, Btest = train_test_split(kbr['title']. values. astype('U'), z, test_size=0.20, random_state=0)
#---------------------PERFORMING COUNT VECTORIZIER OPERATIONS-----------------#
count_vectorizer = CountVectorizer(stop_words='english')
count_train = count_vectorizer.fit_transform(Ktrain)
count_test = count_vectorizer.transform(Ktest)
#-----------------------PERFORMING TFIDF VECTORIZIER OPERATIONS----------------#
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
tfidf_train = tfidf_vectorizer.fit_transform(Ktrain)
tfidf_test = tfidf_vectorizer.transform(Ktest)
#------------------------------------HASH VECTORIZIER----------------------#
hash_vectorizer = HashingVectorizer(stop_words='english')
hash_train = hash_vectorizer.fit_transform(Ktrain)
hash_test = hash_vectorizer.transform(Ktest)
#-----------------------PROGRAM FOR PLOTTING OF CONFUSION MATRIX--------#
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('True label')
    plt.xlabel('Predicted label')

#------------------------------------MULTINOMIAL BAIYES USING TFIDF-------------------#
    Alg = MultinomialNB()
    Alg.fit(tfidf_train, Btrain)
pred = Alg.predict(tfidf_test)
score1 = metrics.accuracy_score(Btest, pred)
pre1 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec1= metrics.recall_score(Btest, pred,pos_label='FAKE')
f11 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score1)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)
```

```
#------------------------------MULTINOMIAL BAIYES USING COUNT-----------------#

Alg = MultinomialNB()
 Alg.fit(count_train, Btrain)
 pred = Alg.predict(count_test)
score2 = metrics.accuracy_score(Btest, pred)
pre2 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec2= metrics.recall_score(Btest, pred,pos_label='FAKE')
f12 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score2)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)

#----------------------PASSIVE AGGRESSIVE CLASSIFIER USING TFIDF---------------#
Alg1 = PassiveAggressiveClassifier()
 Alg1.fit(tfidf_train, Btrain)
pred = Alg1.predict(tfidf_test)
score3 = metrics.accuracy_score(Btest, pred)
pre3 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec3= metrics.recall_score(Btest, pred,pos_label='FAKE')
f3 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score3)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)
   #------------PASSIVE AGGRESSIVE CLASSIFIER USING HASH VECTORIZIER-----#

Alg1 = PassiveAggressiveClassifier()
 Alg1.fit(hash_train, Btrain)
pred = Alg1.predict(hash_test)
score4 = metrics.accuracy_score(Btest, pred)
```

```
pre4 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec4= metrics.recall_score(Btest, pred,pos_label='FAKE')
f14 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score4)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)
#-------------------XGBOOST CLASSIFIER USING THE TFIDF VECTORIZIER------------#
Alg2 = XGBClassifier()
 Alg2.fit(tfidf_train, Btrain)
 pred = Alg2.predict(tfidf_test)
score5 = metrics.accuracy_score(Btest, pred)
pre5 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec5= metrics.recall_score(Btest, pred,pos_label='FAKE')
f15 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score5)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)
#-----------------XGBOOST CLASSIFIER USING THE COUNT VECTORIZIER------------#
Alg2 = XGBClassifier()
 Alg2.fit(count_train, Btrain)
 pred = Alg2.predict(count_test)
score6 = metrics.accuracy_score(Btest, pred)
pre6 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec6= metrics.recall_score(Btest, pred,pos_label='FAKE')
f16 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score6)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)
```

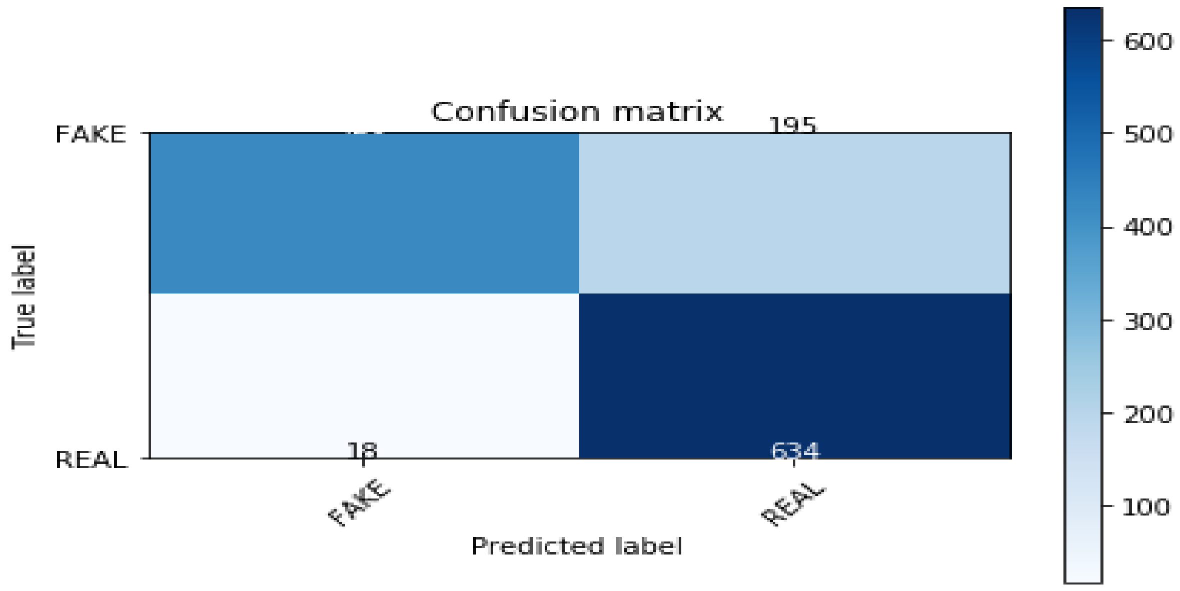
```
#----------------- VISUALIZATION OF THE RECALL VALUE-----------------------#
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
data = [rec1,rec3,rec5]
data2 = ['NB' , 'PGC' , 'XGB']
ax.bar(data2,data)
plt.title( 'recall comparison for tfidf algorithms')
plt.xlabel('Algorithms')
plt.ylabel('value')
plt.show()
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
data = [rec2,rec4,rec6]
data2 = ['NB' , 'PGC' , 'XGB']
ax.bar(data2,data)
plt.title( 'recall comparison for count algorithms')
plt.xlabel('Algorithms')
plt.ylabel('value')
plt.show()
#-------------------------EXTRACTING AN ARTICLE FROM NEWSPAPER-----------------#

from newspaper import Article
url ="####COPY AND PASTE THE URL OF NEWS ARTICLE ####"
article = Article(url, language="en")
article.download()
article.parse()
article.nlp()
text= article.summary
#---------------FINDING FAKE OR REAL USING MNB COUNTVECTORIZIER------------#
```

```
count_test2 = count_vectorizer.transform([text])
Alg.fit(count_train, Btrain)
pred = Alg.predict(count_test2)
#---------------FINDING FAKE OR REAL USING MNB  TFIDF VECTORIZIER----------#
tfidf_test2 = tfidf_vectorizer.transform([text])
Alg.fit(tfidf_train, Btrain)
pred2 = Alg.predict(tfidf_test2)
#---------------FINDING FAKE OR REAL USING PAC  TFIDF VECTORIZIER----------#
tfidf_test2 = tfidf_vectorizer.transform([text])
Alg1.fit(tfidf_train, Btrain)
pred3 = Alg.predict(tfidf_test2)
#---------------FINDING FAKE OR REAL USING HASHVECTORIZIER----------#
hash_test2 = hash_vectorizer.transform([text])
Alg1.fit(hash_train, Btrain)
pred4 = Alg1.predict(hash_test2)
#---------------FINDING FAKE OR REAL USING  XGB COUNT VECTORIZIER---------#
count_test2 = count_vectorizer.transform([text])
Alg2.fit(count_train, Btrain)
pred5 = Alg.predict(count_test2)

#----------------FINDING FAKE OR REAL USING  XGB TFIDF VECTORIZIER----#
tfidf_test2 = tfidf_vectorizer.transform([text])
Alg.fit(tfidf_train, Btrain)
pred6 = Alg.predict(tfidf_test2)
```

CHAPTER 6

RESULTS AND ANALYSIS

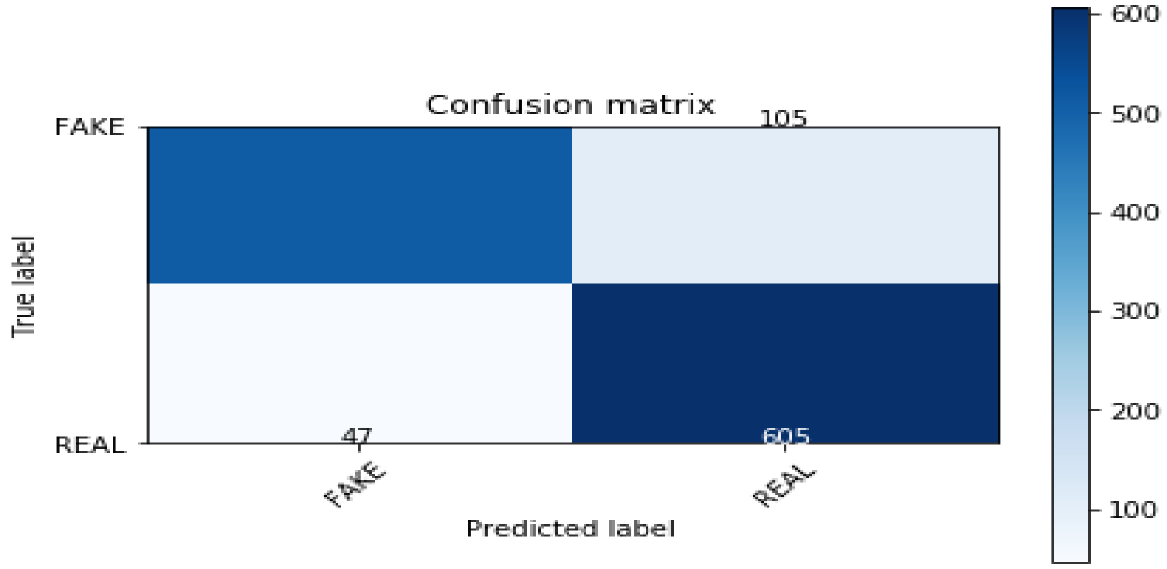
6.1 RESULTS:



[[420 195]

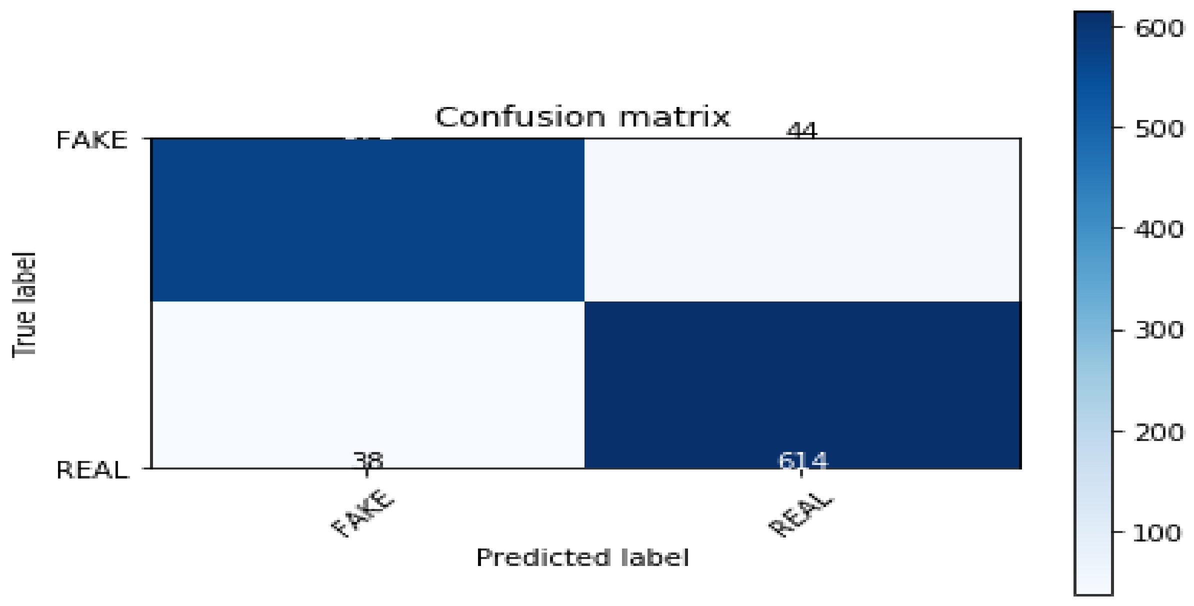
[18 634]]

Fig 1.9 CM for NB tfidf



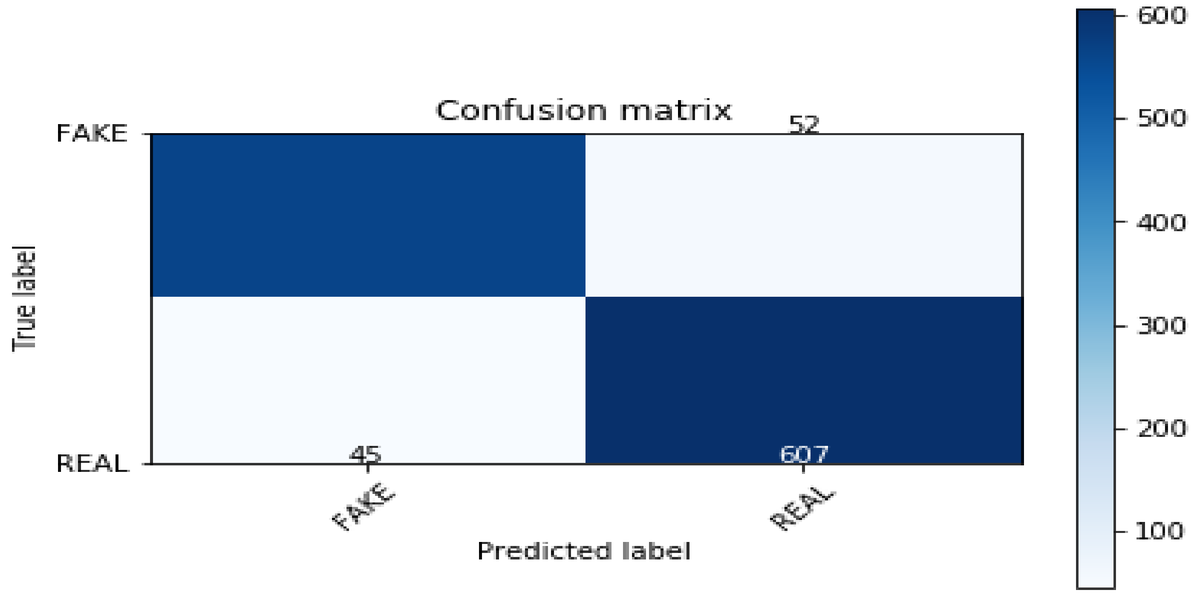
[[510 105]

Fig 1.10 CM for NB count



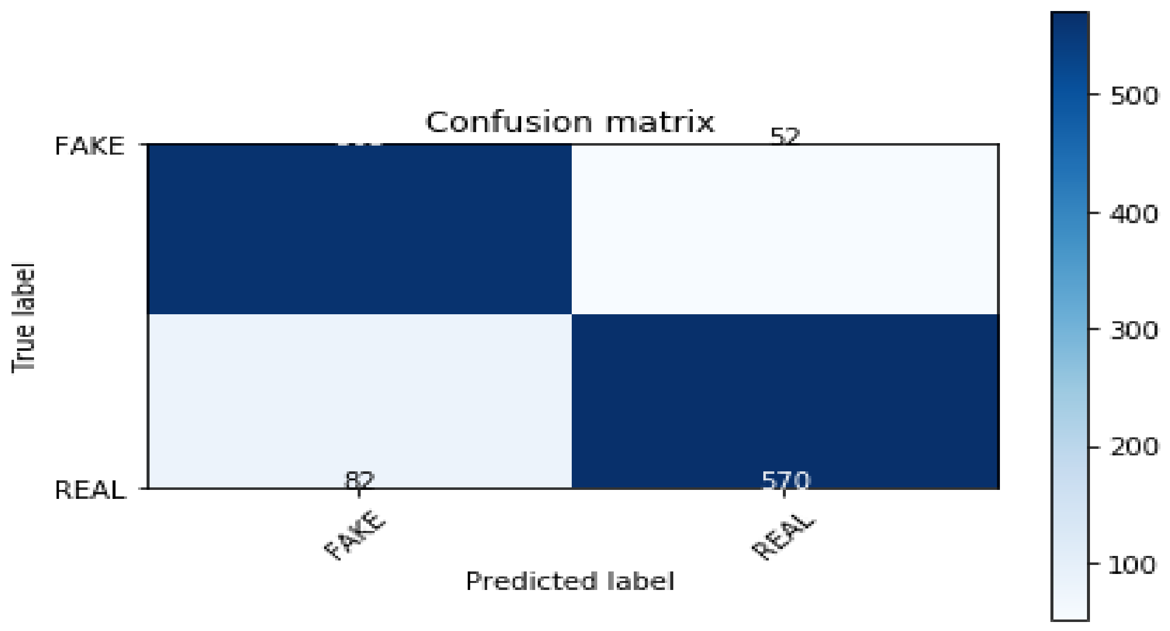
[[571 44]] Fig 1.11 CM for PAC tfidf

[38 614]]



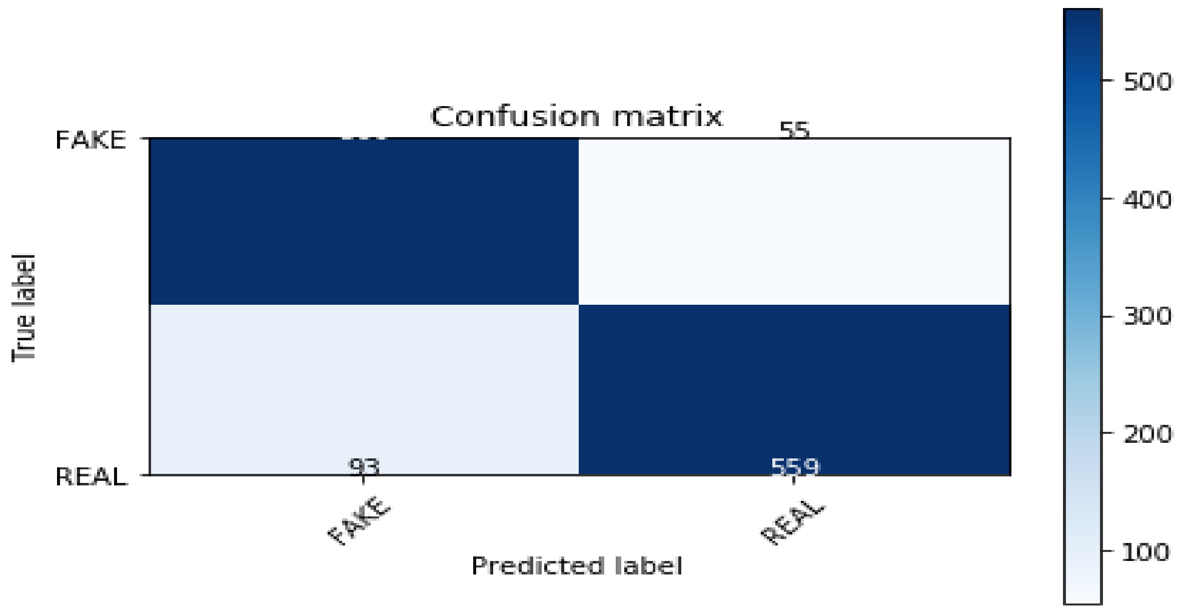
[[563 52]] Fig 1.12 CM for PAC count

[45 607]]



[[563 52]   Fig 1.13   CM for XGB tfidf

 [82 570]]



[[560 55]   Fig 1.14   CM for XGB count

 [93 559]]

6.2 ANALYSIS OF RESULTS

Accuracy	Precision	Recall	F1 score
0.83	0.95	0.68	0.79

Table 3 Multinomial naïve Bayes for Tfidf values

Accuracy	Precision	Recall	F1 score
0.88	0.91	0.82	0.87

Table 4 Multinomial naïve Bayes for Count values

Accuracy	Precision	Recall	F1 score
0.93	0.93	0.93	0.93

Table 5 Passive Aggressive classifier for Tfidf values

Accuracy	Precision	Recall	F1 score
0.92	0.92	0.92	0.92

Table 6 Passive Aggressive classifier for count values

Accuracy	Precision	Recall	F1 score
0.89	0.87	0.91	0.89

Table 7 XgBoost classifier for count values

Accuracy	Precision	Recall	F1 score
0.88	0.85	0.91	0.88

Table 8 XgBoost classifier for count values

From the confusion matrix obtained we have analysed the results and calculated the Accuracy, Precision, Recall and F1 score.

As, we have explained previously for fake news detection the model with better recall is more efficient than the Accuracy or precision.

So, we can infer that Xgboost algorithm is more useful and efficient in finding the fake news.

For more detailed explanation we have represented graphically.

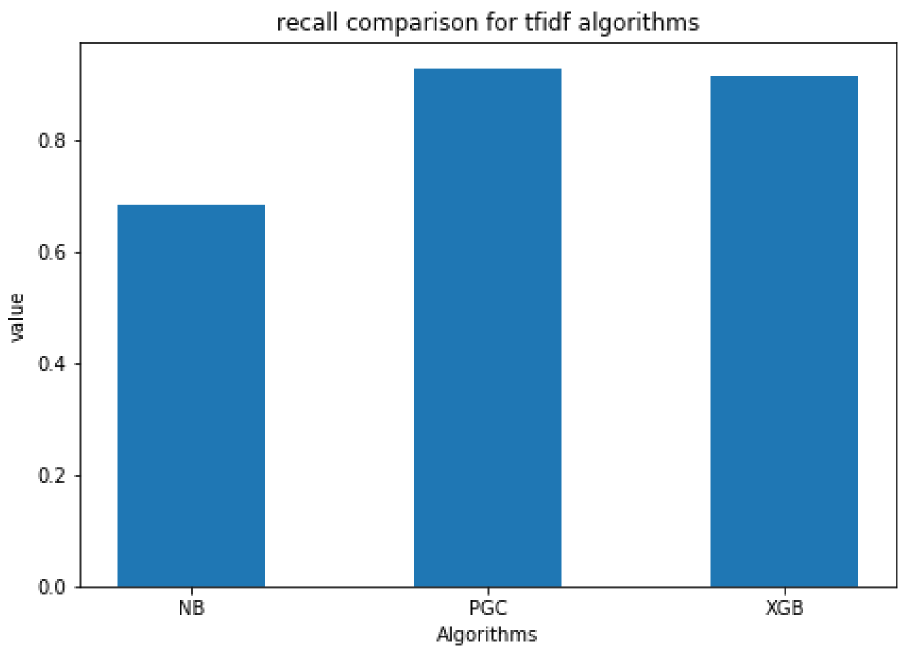


Fig 1.15 Recall comparison of tfidf algorithms

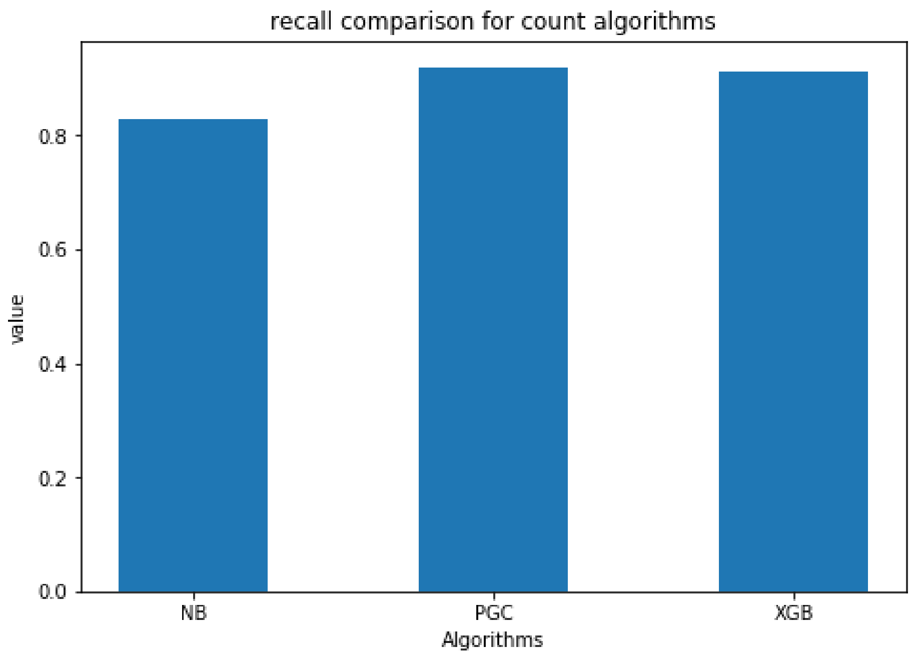
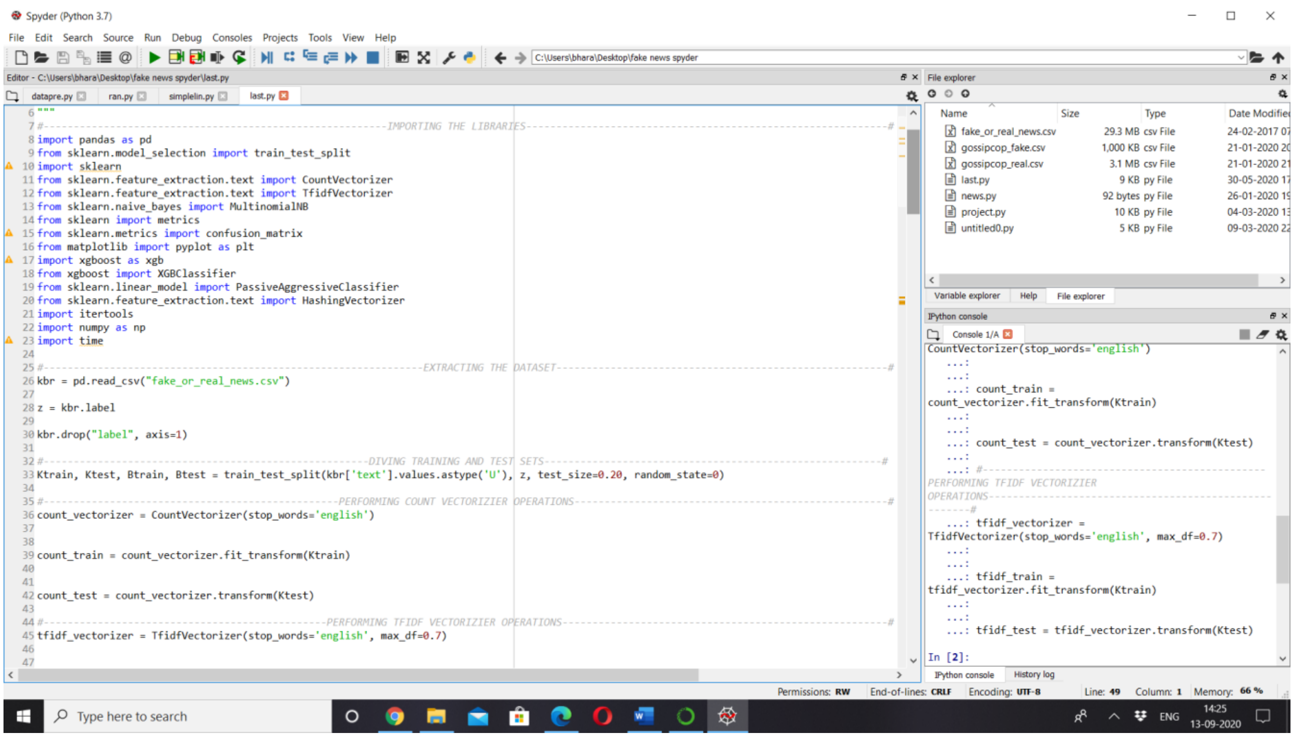


Fig 1.16 Recall comparison of count algorithms

CHAPTER 7

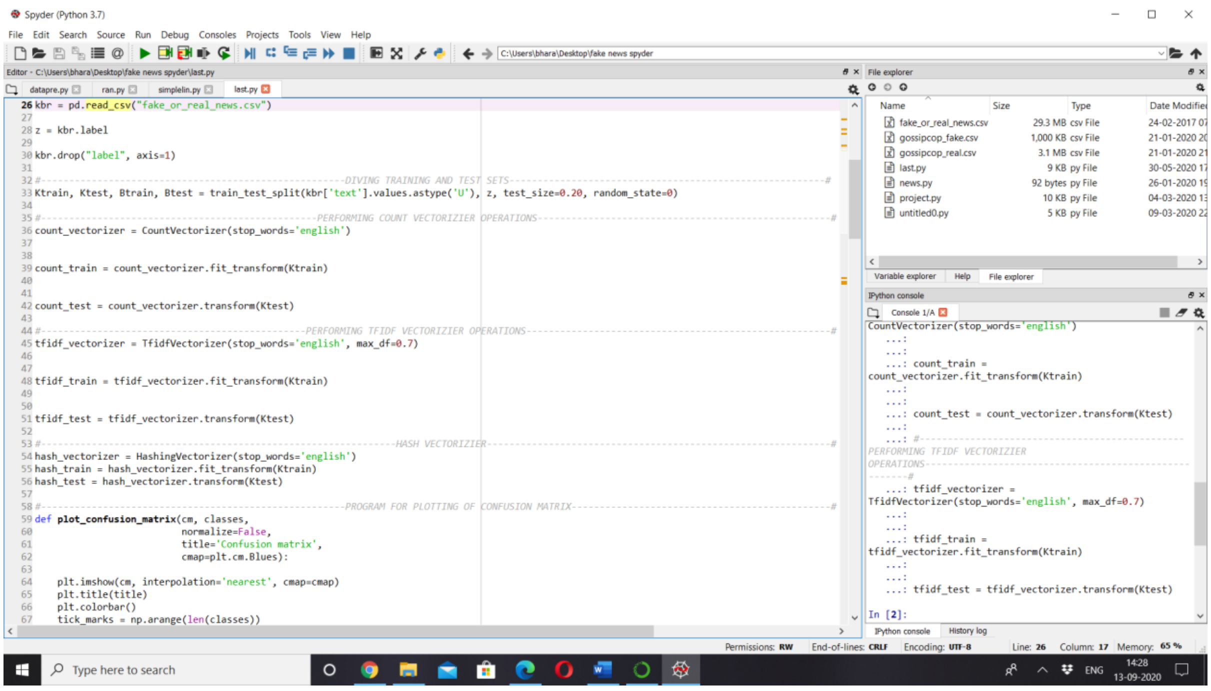
SCREENSHOTS



```
"""
#----------------------------------------------IMPORTING THE LIBRARIES-----------------------------------------------#
import pandas as pd
from sklearn.model_selection import train_test_split
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.feature_extraction.text import HashingVectorizer
import itertools
import numpy as np
import time

#---------------------------------------------------EXTRACTING THE DATASET-------------------------------------------#
kbr = pd.read_csv("fake_or_real_news.csv")

z = kbr.label

kbr.drop("label", axis=1)

#--------------------------------------------DIVING TRAINING AND TEST SETS-------------------------------------------#
Ktrain, Ktest, Btrain, Btest = train_test_split(kbr['text'].values.astype('U'), z, test_size=0.20, random_state=0)

#---------------------------------------PERFORMING COUNT VECTORIZIER OPERATIONS--------------------------------------#
count_vectorizer = CountVectorizer(stop_words='english')


count_train = count_vectorizer.fit_transform(Ktrain)


count_test = count_vectorizer.transform(Ktest)

#--------------------------------------PERFORMING TFIDF VECTORIZIER OPERATIONS---------------------------------------#
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)


```

```
CountVectorizer(stop_words='english')
   ...:
   ...:
   ...: count_train =
count_vectorizer.fit_transform(Ktrain)
   ...:
   ...:
   ...: count_test = count_vectorizer.transform(Ktest)
   ...:
   ...: #---------------------------------------
PERFORMING TFIDF VECTORIZIER
OPERATIONS---------------------------------------
-------#
   ...: tfidf_vectorizer =
TfidfVectorizer(stop_words='english', max_df=0.7)
   ...:
   ...:
   ...: tfidf_train =
tfidf_vectorizer.fit_transform(Ktrain)
   ...:
   ...:
   ...: tfidf_test = tfidf_vectorizer.transform(Ktest)

In [2]:
```

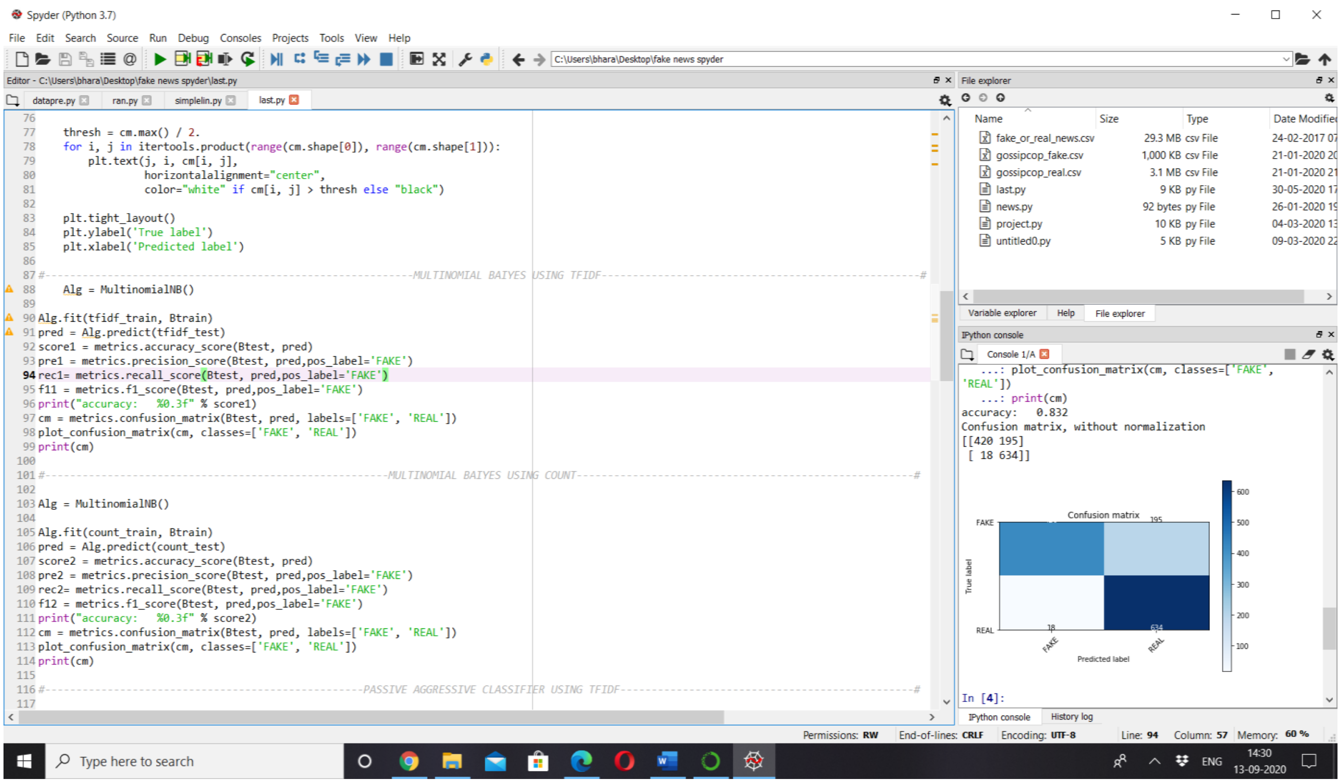
Fig:1.17   Importing and Dividing the data



```
kbr = pd.read_csv("fake_or_real_news.csv")

z = kbr.label

kbr.drop("label", axis=1)

#--------------------------------------------DIVING TRAINING AND TEST SETS-------------------------------------------#
Ktrain, Ktest, Btrain, Btest = train_test_split(kbr['text'].values.astype('U'), z, test_size=0.20, random_state=0)

#---------------------------------------PERFORMING COUNT VECTORIZIER OPERATIONS--------------------------------------#
count_vectorizer = CountVectorizer(stop_words='english')


count_train = count_vectorizer.fit_transform(Ktrain)


count_test = count_vectorizer.transform(Ktest)

#--------------------------------------PERFORMING TFIDF VECTORIZIER OPERATIONS---------------------------------------#
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)


tfidf_train = tfidf_vectorizer.fit_transform(Ktrain)


tfidf_test = tfidf_vectorizer.transform(Ktest)

#---------------------------------------------------HASH VECTORIZIER-------------------------------------------------#
hash_vectorizer = HashingVectorizer(stop_words='english')
hash_train = hash_vectorizer.fit_transform(Ktrain)
hash_test = hash_vectorizer.transform(Ktest)

#-----------------------------------------PROGRAM FOR PLOTTING OF CONFUSION MATRIX------------------------------------#
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))
```

```
CountVectorizer(stop_words='english')
   ...:
   ...:
   ...: count_train =
count_vectorizer.fit_transform(Ktrain)
   ...:
   ...:
   ...: count_test = count_vectorizer.transform(Ktest)
   ...:
   ...: #---------------------------------------
PERFORMING TFIDF VECTORIZIER
OPERATIONS---------------------------------------
-------#
   ...: tfidf_vectorizer =
TfidfVectorizer(stop_words='english', max_df=0.7)
   ...:
   ...:
   ...: tfidf_train =
tfidf_vectorizer.fit_transform(Ktrain)
   ...:
   ...:
   ...: tfidf_test = tfidf_vectorizer.transform(Ktest)

In [2]:
```

Fig1.18   Performing vectorization methods

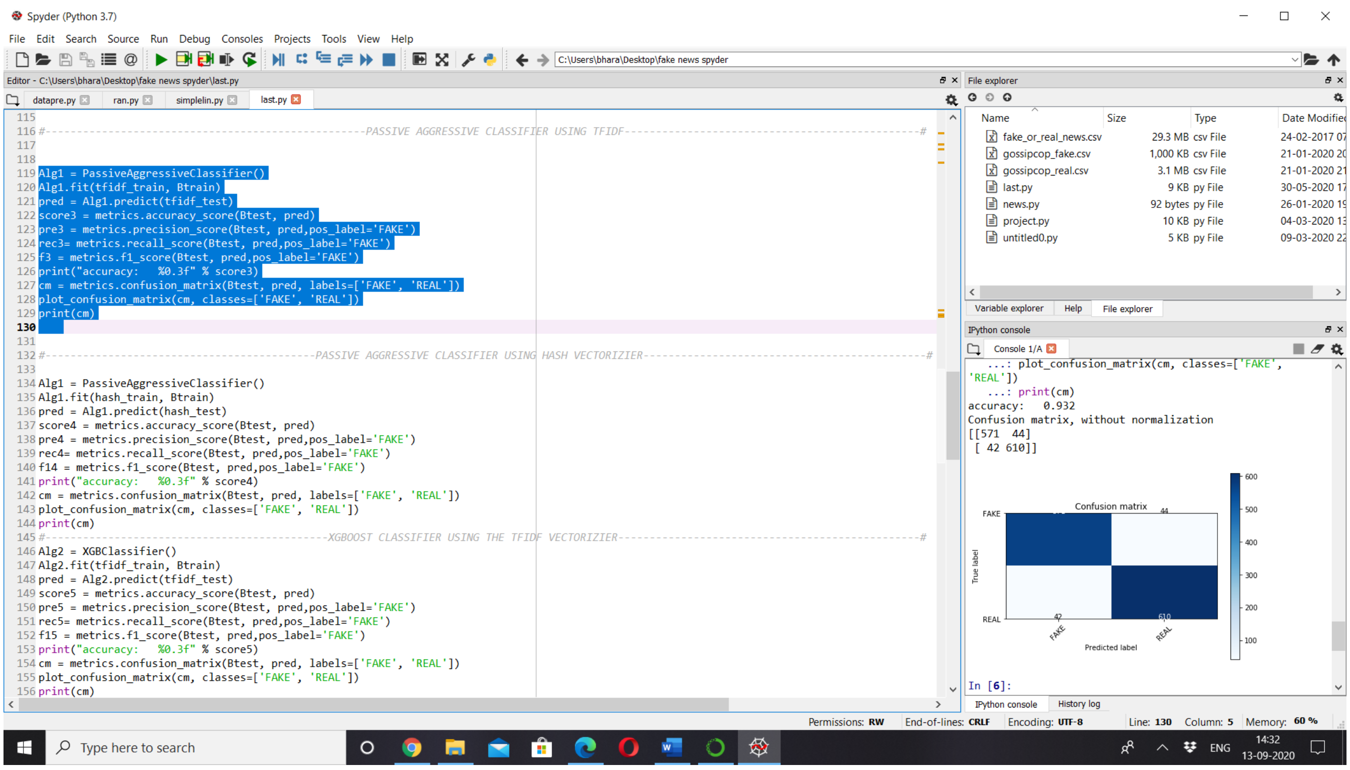


```
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

#-------------------------------------------------MULTINOMIAL BAIYES USING TFIDF--------------------------------#
    Alg = MultinomialNB()

Alg.fit(tfidf_train, Btrain)
pred = Alg.predict(tfidf_test)
score1 = metrics.accuracy_score(Btest, pred)
pre1 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec1= metrics.recall_score(Btest, pred,pos_label='FAKE')
f11 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score1)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)

#-----------------------------------------------MULTINOMIAL BAIYES USING COUNT--------------------------------------#

Alg = MultinomialNB()

Alg.fit(count_train, Btrain)
pred = Alg.predict(count_test)
score2 = metrics.accuracy_score(Btest, pred)
pre2 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec2= metrics.recall_score(Btest, pred,pos_label='FAKE')
f12 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score2)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)

#---------------------------------------------PASSIVE AGGRESSIVE CLASSIFIER USING TFIDF-------------------------------#
```

```
   ...: plot_confusion_matrix(cm, classes=['FAKE',
'REAL'])
   ...: print(cm)
accuracy:   0.832
Confusion matrix, without normalization
[[420 195]
 [ 18 634]]
```

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\bhara\Desktop\fake news spyder
Editor - C:\Users\bhara\Desktop\fake news spyder\last.py
datapre.py
ran.py
simplelin.py
last.py
File explorer
Name
Size
Type
Date Modified
fake_or_real_news.csv 29.3 MB csv File 24-02-2017
gossipcop_fake.csv 1,000 KB csv File 21-01-2020
gossipcop_real.csv 3.1 MB csv File 21-01-2020
last.py 9 KB py File 30-05-2020
news.py 92 bytes py File 26-01-2020
project.py 10 KB py File 04-03-2020
untitled0.py 5 KB py File 09-03-2020
Variable explorer
Help
File explorer
IPython console
Console 1/A
Confusion matrix
FAKE
REAL
True label
Predicted label
195
18
634
600
500
400
300
200
100
In [4]:
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: UTF-8
Line: 94
Column: 57
Memory: 60 %
Type here to search
ENG
14:30
13-09-2020

Fig:1.19 Multinomial Bayes Implementation

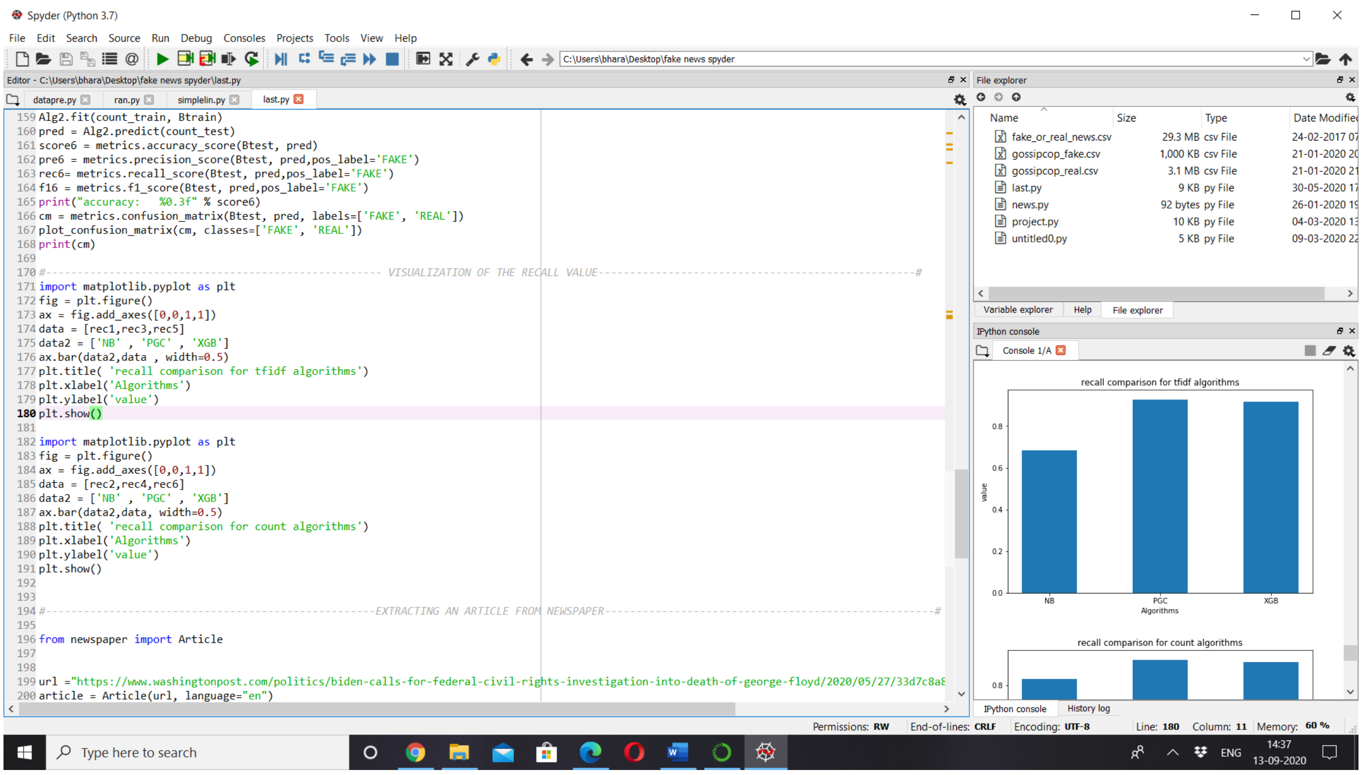


```
#---------------------------------------------PASSIVE AGGRESSIVE CLASSIFIER USING TFIDF-------------------------------#


Alg1 = PassiveAggressiveClassifier()
Alg1.fit(tfidf_train, Btrain)
pred = Alg1.predict(tfidf_test)
score3 = metrics.accuracy_score(Btest, pred)
pre3 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec3= metrics.recall_score(Btest, pred,pos_label='FAKE')
f3 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score3)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)


#-----------------------------------PASSIVE AGGRESSIVE CLASSIFIER USING HASH VECTORIZIER------------------------------------#

Alg1 = PassiveAggressiveClassifier()
Alg1.fit(hash_train, Btrain)
pred = Alg1.predict(hash_test)
score4 = metrics.accuracy_score(Btest, pred)
pre4 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec4= metrics.recall_score(Btest, pred,pos_label='FAKE')
f14 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score4)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)
#---------------------------------------XGBOOST CLASSIFIER USING THE TFIDF VECTORIZIER---------------------------------------#
Alg2 = XGBClassifier()
Alg2.fit(tfidf_train, Btrain)
pred = Alg2.predict(tfidf_test)
score5 = metrics.accuracy_score(Btest, pred)
pre5 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec5= metrics.recall_score(Btest, pred,pos_label='FAKE')
f15 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score5)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)
```

```
   ...: plot_confusion_matrix(cm, classes=['FAKE',
'REAL'])
   ...: print(cm)
accuracy:   0.932
Confusion matrix, without normalization
[[571  44]
 [ 42 610]]
```

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\bhara\Desktop\fake news spyder
Editor - C:\Users\bhara\Desktop\fake news spyder\last.py
datapre.py
ran.py
simplelin.py
last.py
File explorer
Name
Size
Type
Date Modified
fake_or_real_news.csv 29.3 MB csv File 24-02-2017
gossipcop_fake.csv 1,000 KB csv File 21-01-2020
gossipcop_real.csv 3.1 MB csv File 21-01-2020
last.py 9 KB py File 30-05-2020
news.py 92 bytes py File 26-01-2020
project.py 10 KB py File 04-03-2020
untitled0.py 5 KB py File 09-03-2020
Variable explorer
Help
File explorer
IPython console
Console 1/A
Confusion matrix
FAKE
REAL
True label
Predicted label
44
42
610
600
500
400
300
200
100
In [6]:
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: UTF-8
Line: 130
Column: 5
Memory: 60 %
Type here to search
ENG
14:32
13-09-2020

Fig:1.20 PASSIVE AGGRESSIVE CLASSIFIER IMPLEMENTATION


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\bhara\Desktop\fake news spyder
Editor - C:\Users\bhara\Desktop\fake news spyder\last.py
datapre.py
ran.py
simplelin.py
last.py
File explorer
Variable explorer
Help
File explorer
IPython console
Console 1/A
recall comparison for tfidf algorithms
value
NB
PGC
XGB
Algorithms
recall comparison for count algorithms
IPython console
History log
Type here to search
ENG
14:37
13-09-2020

```
Alg2.fit(count_train, Btrain)
pred = Alg2.predict(count_test)
score6 = metrics.accuracy_score(Btest, pred)
pre6 = metrics.precision_score(Btest, pred,pos_label='FAKE')
rec6= metrics.recall_score(Btest, pred,pos_label='FAKE')
f16 = metrics.f1_score(Btest, pred,pos_label='FAKE')
print("accuracy:   %0.3f" % score6)
cm = metrics.confusion_matrix(Btest, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(cm)

#----------------------------------------------------- VISUALIZATION OF THE RECALL VALUE-----------------------------------------------------#
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
data = [rec1,rec3,rec5]
data2 = ['NB' , 'PGC' , 'XGB']
ax.bar(data2,data , width=0.5)
plt.title( 'recall comparison for tfidf algorithms')
plt.xlabel('Algorithms')
plt.ylabel('value')
plt.show()

import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
data = [rec2,rec4,rec6]
data2 = ['NB' , 'PGC' , 'XGB']
ax.bar(data2,data, width=0.5)
plt.title( 'recall comparison for count algorithms')
plt.xlabel('Algorithms')
plt.ylabel('value')
plt.show()


#------------------------------------------------------EXTRACTING AN ARTICLE FROM NEWSPAPER------------------------------------------------------#

from newspaper import Article


url ="https://www.washingtonpost.com/politics/biden-calls-for-federal-civil-rights-investigation-into-death-of-george-floyd/2020/05/27/33d7c8a8
article = Article(url, language="en")
```

Name	Size	Type	Date Modified
fake_or_real_news.csv	29.3 MB	csv File	24-02-2017 07
gossipcop_fake.csv	1,000 KB	csv File	21-01-2020 20
gossipcop_real.csv	3.1 MB	csv File	21-01-2020 21
last.py	9 KB	py File	30-05-2020 17
news.py	92 bytes	py File	26-01-2020 19
project.py	10 KB	py File	04-03-2020 13
untitled0.py	5 KB	py File	09-03-2020 22

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 180 Column: 11 Memory: 60 %

Fig:1.21 PLOTTING THE GRAPHS

CHAPTER 8

8.1 CONCLUSION

Fake news detection is the process of distinguishing of fake news from real news in this process we have used machine learning algorithms on a specifically created political dataset to test our algorithms. we have used algorithms such as Multinomial Naive Bayes, Passive Aggressive classifier and Xgboost algorithms in the combination of the count and term frequency(tfidf). From the results obtained we can say that the Xgboost algorithm is most useful and efficient in finding of the fake news because of the large amount of recall value compared to the other algorithms .Because avoiding the spread of fake news is more important than spreading of real news recall is most important in this model .But still there is no machine learning model which can accurately distinguish fake or real news it requires large of amount diversified datasets and combination of various algorithms .

8.2 Future Enhancement:

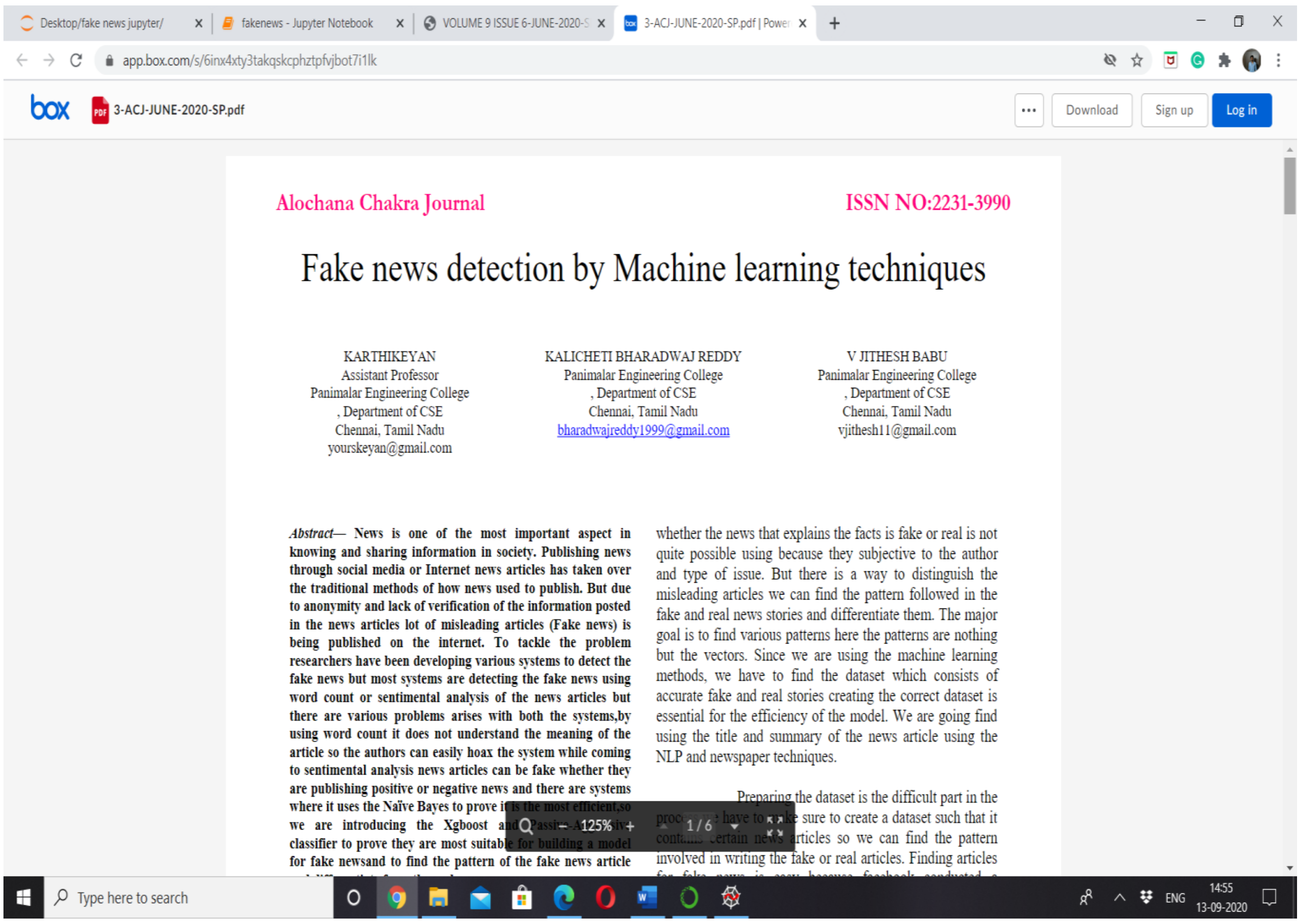
In future models we will incorporate a greater number of datasets in more fields for differentiating the fake and real news and we are going to detect the fake news in images and video.

APPENDIX

Journal Publication link and Screenshot:

Link: http://www.alochanachakra.in/VOLUME-9-ISSUE-6-JUNE-2020-SP-1/

Screenshot:


Desktop/fake news jupyter/
fakenews - Jupyter Notebook
VOLUME 9 ISSUE 6-JUNE-2020-S
3-ACJ-JUNE-2020-SP.pdf | Power
app.box.com/s/6inx4xty3takqskcphztpfvjbot7i1lk
box
3-ACJ-JUNE-2020-SP.pdf
Download
Sign up
Log in
125%
1 / 6
Type here to search
ENG
14:55
13-09-2020

Alochana Chakra Journal ISSN NO:2231-3990

Fake news detection by Machine learning techniques

KARTHIKEYAN
Assistant Professor
Panimalar Engineering College
, Department of CSE
Chennai, Tamil Nadu
yourskeyan@gmail.com

KALICHETI BHARADWAJ REDDY
Panimalar Engineering College
, Department of CSE
Chennai, Tamil Nadu
bharadwajreddy1999@gmail.com

V JITHESH BABU
Panimalar Engineering College
, Department of CSE
Chennai, Tamil Nadu
vjithesh11@gmail.com

Abstract— **News is one of the most important aspect in knowing and sharing information in society. Publishing news through social media or Internet news articles has taken over the traditional methods of how news used to publish. But due to anonymity and lack of verification of the information posted in the news articles lot of misleading articles (Fake news) is being published on the internet. To tackle the problem researchers have been developing various systems to detect the fake news but most systems are detecting the fake news using word count or sentimental analysis of the news articles but there are various problems arises with both the systems,by using word count it does not understand the meaning of the article so the authors can easily hoax the system while coming to sentimental analysis news articles can be fake whether they are publishing positive or negative news and there are systems where it uses the Naïve Bayes to prove it is the most efficient,so we are introducing the Xgboost and Passive-Aggressive classifier to prove they are most suitable for building a model for fake newsand to find the pattern of the fake news article**

whether the news that explains the facts is fake or real is not quite possible using because they subjective to the author and type of issue. But there is a way to distinguish the misleading articles we can find the pattern followed in the fake and real news stories and differentiate them. The major goal is to find various patterns here the patterns are nothing but the vectors. Since we are using the machine learning methods, we have to find the dataset which consists of accurate fake and real stories creating the correct dataset is essential for the efficiency of the model. We are going find using the title and summary of the news article using the NLP and newspaper techniques.

Preparing the dataset is the difficult part in the process we have to make sure to create a dataset such that it contains certain news articles so we can find the pattern involved in writing the fake or real articles. Finding articles

REFERENCES

[1] Conroy, N. J., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. Proceedings of the Association for Information Science and Technology,

[2] B. Bhutani, N. Rastogi, P. Sehgal and A. Purwar, "Fake News Detection Using Sentiment Analysis," *2019 Twelfth International Conference on Contemporary Computing (IC3)*, Noida, India, 2019, pp. 1-5, doi: 10.1109/IC3.2019.8844880.

[3] C. K. Hiramath and G. C. Deshpande, "Fake News Detection Using Deep Learning Techniques," *2019 1st International Conference on Advances in Information Technology (ICAIT)*, Chikmagalur, India, 2019, pp. 411-415, doi: 10.1109/ICAIT47043.2019.8987258.

[4] R. K. Kaliyar, "Fake News Detection Using A Deep Neural Network," 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018, pp. 1-7, doi: 10.1109/CCAA.2018.8777343.

[5] Z. I. Mahid, S. Manickam and S. Karuppayah, "Fake News on Social Media: Brief Review on Detection Techniques," 2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA), Subang Jaya, Malaysia, 2018, pp. 1-5, doi: 10.1109/ICACCAF.2018.8776689.

[6] Alcott, Hunt, and Matthew Gentzkow. 2017. "Social Media and Fake News in the 2016 Election." *Journal of Economic Perspectives*, 31 (2): 211-36.