Noname manuscript No.
(will be inserted by the editor)

Automating Fake News Detection System using Multi-level Voting Model

Sawinder Kaur · Parteek Kumar · Ponnurangam Kumaraguru

Received: date / Accepted: date

Abstract The issues of online fake news have attained an increasing eminence in the diffusion of shaping news stories online. Misleading or unreliable information in form of videos, posts, articles, URLs are extensively disseminated through popular social media platforms such as Facebook, Twitter, *etc.* As a result, editors and journalists are in need of new tools that can help them to pace up the verification process for the content that has been originated from social media. Motivated by the need for automated detection of fake news, the goal is to find out which classification model identifies phony features accurately using three feature extraction techniques, Term Frequency-Inverse Document Frequency (Tf-Idf), Count-Vectorizer (CV) and Hashing-Vectorizer (HV). Also, in this paper, a novel multi-level voting ensemble model is proposed. The proposed system has been tested on three datasets using twelve classifiers. These ML classifiers are combined based on their false prediction ratio. It has been observed that the Passive Aggressive (PA), Logistic Regression (LR) and Linear Support Vector (LinearSVC) individually performs best using TF-IDF, CV and HV feature extraction approaches, respectively based on their performance metrics. Whereas, the proposed model outperforms the Passive Aggressive model by 0.8%, Logistic Regression model by 1.3%, LinearSVC model by 0.4% using TF-IDF, CV and HV, respectively. The proposed system can also be used to predict the fake content (textual form) from online social media websites.

Keywords Fake news articles, Count-Vectorizer, Tf-Idf, Hashing-Vectorizer, Classifiers, Textual content, Machine learning models

Sawinder Kaur (Corresponding Author)
Doctoral Research Lab-II,
Computer Science and Engineering Department,
TIET, Patiala, India.
Tel.: +91-8837761601
E-mail: sawinderkaurvohra@gmail.com

Parteek Kumar
Computer Science and Engineering Department,
TIET, Patiala, India.
E-mail: parteek.bhatia@thapar.edu

Ponnurangam Kumaraguru
Computer Science and Engineering Department,
IIIT, Delhi, India.
E-mail: pk@iiitd.ac.in

1 Introduction

A growing interest related to fake news detection has attracted many researchers as fake information is circulated through online social media platforms such as Facebook, Twitter, *etc.* The fake content is spreading at a faster pace to gain popularity over social media, to distract people from the current critical issues. Most of the people believe that the information they receive from various social media sites is reliable and true, *i.e.*, people are inherently truth-biased. Also, people easily trust and want to believe in what they actually interpret in their minds, *i.e.*, confirmation-biased. In general, it has been analyzed that humans are unable to recognize deception effectively. Due to which a serious and negative impact of fake articles can be seen on society and individuals leading to an imbalance of the news ecosystem. It was observed that during US president election [1], most of the widely spread articles on social media were fake. Recently a fake video related to Kerela battling with floods was viral on social media platform (Facebook) claiming that the Chief Minister of the state is forcing the Indian Army to stop conducting the rescue operations in flooded regions of Kerela. Also, during India's national election (2019), various WhatsApp groups (> 900,000) were created to disseminate the fake information regarding India's ruling party [2]. Most of the fake articles are created to confuse people and trigger their distrust. Such problems led researchers to look at some automated ways to access the ground truth values of fake text on the basis of the textual content posted in articles on social platforms.

Social media enables to maintain and develop relations with others. It helps the users to present themselves by creating their profiles, sharing information through photos, images, and text to link with other members [3]. Some of the most popular social media [4] sites are Facebook, Twitter [5] [6] [7], Instagram [8], WhatsApp [9] [10], LinkedIn, WeChat, Snapchat, Foursquare [11]. With the popularity of social media sites [12], level of usage to share content on online social media has increased. There are several reasons for the change in behavior for such kind of consumptions. The content shared on social media platforms requires less time and cost than on newspapers or traditional news media. Easier to share content in the form of video, blogs, posts with friends or users. This gives the growth ease to the authors and publishers to publish their contents as articles on collaborative environments. There is 13% of the global increase in social media usage since 2017 [9]. Distribution and creation of news content in the form of posts [13], blogs, articles, images, videos, *etc.*, have been spreading through social media websites [12]. This increase in social media also gives rise in the spread of fake articles [14] over the internet.

1.1 Types of Fake News

According to the literature, there are five types of fakes news. The first type can be seen in the form of **deliberate misinformation**, which is misleading information that is spread in a calculated way to deceive targeted users. Other forms of fake news can be **clickbait** [15] [16] which grab the reader's attention with a purpose to make them click on the fake news seen on the internet. Users who set up fake sites generate huge revenues and clicking on such websites results in bombarded ads. Articles [14] from satirical sources like 'The Onion' often repeat and share the news as if the printed stories were true. **Parody or Satirical** [17] articles use obscenity, absurdity and exaggeration to comment on current events and to unease the readers. **False headlines** are intentionally exaggerated to draw the reader's attention. In such headlines, the title of the articles may not match the context of stories, the headline can be read as one way and state something different as a fact. This type of fake news is untrue at worst and misleading at best. **Hoaxes** is another type of misinformation which deceives the reader deliberately by causing harm and material losses to the users.

1.2 Contribution

Researchers have analyzed that an automated system sometimes identifies fake news articles better than human do. The automated systems can play an important tool for identifying fake stories, articles, blogs, clicks which manipulate public opinion on social networking sites [18] [19]. Taking into need for the development of such fake news detection system, in this paper we have identified news articles as fake or real by using supervised machine learning classifiers such as Naïve Bayes (NB), Decision Tree (DT), Support Vector Machine (SVM), Linear models, Neural Networks (NN) and Ensemble models. To get effective results, three different corpora (News Trends, Kaggle and Reuters) have been collected with similar characteristics. The feature extraction techniques, Term Frequency-Inverted Document Frequency (Tf-Idf), Count-Vectorizer (CV), Hashing-Vectorizer (HV) are used to extract feature vectors from the textual content of articles. The effectiveness of a set of classifiers is observed when they predict the labels of the testing samples after learning the training data using these features extraction techniques.

Various supervised ML models are extensively used for categorization of textual data as fake or real, but such models were not able to obtain the results of an ideal classifier. So, a multi-level voting model has been proposed in this paper to build an ideal classifier with significant improvement than previously existing models. Our contribution to this paper is as follows.

- Statistical analysis of collected datasets (News Trends, Kaggle and Reuters) with negative and positive instances has been performed.
- Twelve ML models are evaluated using Tf-Idf, CV, HV feature extraction techniques to retrieve the best model based on performance metrics.
- A novel method is proposed to merge the ML models based on false prediction rate.
- Proposed an ideal multi-level voting classifier (three-level) to verify the effectiveness of the ensemble model.
- A comparative study is performed to show the effectiveness of our proposed model.

The performance of the multi-level voting system is analyzed using parameters like precision, recall, specificity, ROC curve, F1-score.

The remaining paper is organized as Section 2 gives a brief overview of the related work done in the field of fake news classification. The problem statement is discussed in Section 3. Section 4 covers the methodology of the proposed system. Section 5 introduces the classifiers used for detecting the fake and real articles. The evaluation phase along with analysis done by all classifiers is presented in Section 6. The proposed model is discussed in Section 7 and its performance is evaluated in Section 8. The comparison of the proposed model with existing work is discussed in Section 9. Section 10 concludes the paper along with its future works.

2 Related Work

Many techniques have been developed recently for fake news detection. In this section, the closely related research work that has been done on detecting fake news on online social media is discussed.

Most of the social sites require energy and time to manually remove or filter spam. Markines *et al.* (2009) proposed six highlights (TagSpam, TagBlur, DomFp, NumAds, Plagiarism, ValidLinks) of tagging systems catching diverse properties of social spam [20]. Utilizing the six proposed highlights, creators assessed different administered machine learning techniques to identify spam with precision over 98% with a false positive rate of 2%. The Weka tool used for the experiment gives the best accuracy when evaluated on the basis of Adaboost classifier. To address the issue of detecting video promoters and spammers, Benevenuto *et al.* (2009) manually assembled test gathering of genuine YouTube clients and classified them as legitimates, spammers and promoters. Authors have investigated the feasibility for detecting spammers and promoters by applying a supervised classification algorithm [21]. The classifier used in this paper correctly identifies the majority of promoters correctly.

Qazvinian *et al.* (2011) explored three features such as content-based, network-based and microblog specific memes to correctly identify the rumors [22]. Such features are also used to identify disinformers or users who endorse a rumor and further tries to spread it. For the experiment, authors collected 10,000 manually annotated tweets from twitter and achieved 0.95 in Mean Average Precision (MAP). Rubin *et al.* (2016) proposed a satire detection model with Support Vector Machine (SVM) based algorithm across 4 domains, such as science, business, soft news and civics [17]. To verify the sources of news articles, authors have discussed various legitimate and satirical news websites. In this paper, five features together are chosen to predict the best predicting feature combination with 90% precision and 84% recall to identify satirical news which can help to minimize deception impact of satire.

Analysis for the real event such as Boston Marathon Blasts is done by Gupta *et al.* (2013). During the event, it was observed that a lot of fake and malicious profiles originated on Twitter. Results showed that 29% of the content originated during the Boston Blasts [5] was viral whereas 51% was generic opinions and comments. Authors identified six thousand profiles which were immediately created after the blasts occurred and were suspended by Twitter. Tabloids are often used for sensationalization, exaggeration, producing misleading and low-quality news content. A new form of tabloidization has emerged known as clickbaiting. There exists both non-textual and textual clickbaiting, which is surveyed by Chen *et al.* (2015), who proposed a hybrid approach [15] for automatic detection of clickbait.

Rubin *et al.* (2015) utilized Vector Space Modelling (VSM) and Rhetorical Structure Theory (RST) to analyze misleading and truthful news. RST catches the coherence of story in terms of functional relations among the useful text units and also describe the hierarchical structure for each article/news. VSM is used for identifying the relations among rhetorical structure [23], *i.e.*, each article content can be depicted as vectors in high dimensional space.

Researchers have used different techniques to identify and review the fake content. One of the best and common feature extraction method is Bag of Words. This comprises of a group of words retrieved from the textual content, from where n-gram [4] features can be extracted. The second most important feature which is similar to the Bag of Words approach is Term Frequency (TF) which is related to frequency of the words. Conroy *et al.* (2015) proposed a hybrid approach which combines both machine learning and linguistic cues with network-based behavioral data [1]. The hybrid approach follows both n-gram and bag of word techniques to represent data. Ahmed *et al.* (2017) proposed a fake news detection system which uses n-gram [24] analysis and Term Frequency- Inverse Document Frequency (Tf-Idf) as a feature extraction technique [4]. In this paper, six classifiers of machine learning are used and two different feature extraction techniques are used for comparison and investigation. Volkova *et al.* (2017) built a predictive model to manage 130K news posts as verified or malicious. Authors have classified four subtypes of suspicious news such as propaganda, clickbait, hoaxes and satire [25].

Chhabra *et al.* (2011) had put forward a URL static feature based detection method for detecting malicious websites with accurate results. The author has focussed on external features such as IP addresses. Further, a vector construction VSM [26] is chosen as the URL vector model. The dataset taken in this paper consists of malicious URLs which was downloaded from the phishing platform named as 'Phishtank'[27].

In our digital world, fake news is disseminating and impacting millions of users on social media platforms every day. It has really become difficult to separate truth from fiction. With the help of machine learning models, it is possible to detect spam emails at an early stage with the help of spam filters. ML classifiers help to solve the real world problems. Also, ML has made easier for the users of e-commerce business as it helps to identify the hidden pattern, groups the similar products into a cluster and displays the result to end-user which enables a product based recommendation system. It also helps to solve the problem of unfair recommendations [28].

Comparative analysis of related work done in the field of fake news detection and the proposed system presented in this paper is shown in Table 1.

It has been analyzed that the research work done in the field of fake news detection is mainly restricted to SVM and Naïve Bayes classifiers using only n-gram [24] and Tf-Idf features extraction approaches. No work has been done on Multi-Layer Perceptron (MLP), Long Short term Memory (LSTM) [29] models and hashing based extraction approach which also accounts for better efficiency. Also, the Existing fake news detection [30] models are built using supervised machine learning algorithms whereas manual hand-crafted feature extraction is more time consuming and inefficient method to achieve the best accuracy. Existing techniques studied so far provides a direction to be followed further for quantitative and qualitative research.

3 Problem Statement

To address the issue of fake news generation and dissemination through various online social platforms, an appropriate feature extraction technique is chosen to improve the efficiency of the existing ML classifiers. A novel multi-level voting ensemble model will be proposed to develop an efficient fake news detection system. Mathematically, the problem statement can be represented as- To identify $S = \{fake, real\}$ for a document D where $D = \{t_1, t_2, ..., t_n\}$ and t_i represents the text in a news article a_i chosen from corpus with series of engagements that is composed of title, body and label of the article as $e_{ijk} = (t_i, b_j, l_k)$. The task is to evaluate and analyze the best feature extraction method f_m where $m = \{\text{Tf-Idf, CV, HV}\}$ using machine learning classifier to compute high efficiency in our proposed system. The approach followed in this paper has been discussed in the next Section.

4 Methodology

The architecture of the proposed fake news article detection system [31] is seen in Figure 1. To train the system three corpora has been collected from three different sources by downloading the datasets from News Trends, Kaggle and Reuters websites. In the pre-processing phase, stop words and duplicate text from news articles is removed. The missing values *i.e.*, not available (NA) values are collected and cleaned in the next step. The data retrieved is then split into two parts, training (0.67) and testing (0.33) sets. The feature extraction phase is then carried out to retrieve meaningful features from the textual data. In this phase, the features are extracted from the articles. Three feature extraction techniques such as Term Frequency-Inverted Document Frequency (assigns weights according to the importance of the terms in the document), Count-Vectorizer (counts the frequency of the terms in a document) and Hashing-Vectorizer (follows the hashing trick) have been applied. The features retrieved are then fed to the classification algorithm chosen in next phase. The various ML models such as MultinomialNB, Passive Aggressive, Stochastic Gradient Descent, Logistic Regression, Support Vector Classifier, Nu-Support Vector Classifier, Multi-Layer Perceptron, Linear-Support Vector Classifier, AdaBoost, Gradient Boosting, Decision Tree, Voting classifiers [32] are chosen to learn and identify the patterns and outcomes from them. The models are then evaluated based on performance metrics to achieve an efficient classifier. Based on the analysis done, the models are integrated to propose a multi-level voting model to achieve high efficiency and then is compared with the existing work [33] as discussed in Section 9. The detailed working of each phase, that has been implemented in python framework is discussed below.

4.1 Data Collection

There are many sources of fake article generation such as Facebook [34], Twitter [35] [36] [37] [38], *etc.*, which are used as a trading platform to disseminate fake news. We have used News Trends [39], Kaggle [40] and Reuters [41] dataset with similar attributes such as headlines, body, publisher name of the article, published date, categorical and missing values. The News Trends, Kaggle, and Reuters corpus consist of 7,795; 4,048 and 21578 news articles labeled as fake and real news, respectively.

Table 1: Comparative analysis of research studies

Authors	**Proposed Approach**	**Model**	**Dataset**	**Features**
Markines *et al.* (2009)	Analyzed distinct six features for detecting social spammers using machine learning.	SVM, AdaBoost	Spam posts, tags.	TagSpam, TagBlur, DomFp, NumAds, Plagiarism, ValidLinks
Benevenuto *et al.* (2009)	A video response crawler is proposed to identify spammers in online video social network.	SVM	Real YouTube user information.	Video attributes, individual characteristics of user behavior, social relation between users via video response interactions.
Qazinian *et al.* (2011)	Identified tweets in which rumor is endorsed.	Naïve Bayes	Tweets	Content-based, network-based, Twitter specific memes.
Chhabra *et al.* (2011)	Using URLs static features, a method is developed to detect malicious websites.	Naïve Bayes, Logistic Regression, DT, SVM-RBF, SVM-Linear, SVM-Sigmoid	Malicious URL dataset from 'Phishtank'	Grammar, Lexical, Vectors and Static.
Gupta *et al.* (2013)	Analysis of Twitter content during Boston Marathon.	Logistic Regression	Tweets and corresponding user information	Topic engagement, Global engagement, Social reputation, Likability, Credibility
Chen *et al.* (2015)	Analyzed coherence relations between deceptive and truthful news.	VSM	News samples from NPR's 'Bluff the Listener'	Discourse
Rubin *et al.* (2015)	A hybrid approach is proposed combining linguistic and network-based behavior data.	Linguistic, Network models	Simple text sentences	Bag of Words, n-gram
Conroy *et al.* (2015)	A satire detection model is developed.	SVM	US and Canadian national newspapers	Absurdity, Humor, Grammar, Negative affect, Punctuation.
Ahmed *et al.* (2017)	Developed n-gram based classifier to differentiate between fake ad real articles.	LinearSVM	News articles	TF-IDF
Caetano *et al.* (2018)	A predictive model was built to predict 4 subtypes of suspicious news; satire, hoaxes, clickbait and propaganda.	Linguistic models	News posts	TF-IDF, Doc2Vec
Proposed system	Using textual data of articles, an efficient multi-level voting model is developed to detect fake articles.	SGD, PA, MultinomialNB, Gradient Boosting, DT,AdaBoost	News articles	TF-IDF, CountVectorizer, HashingVectorizer

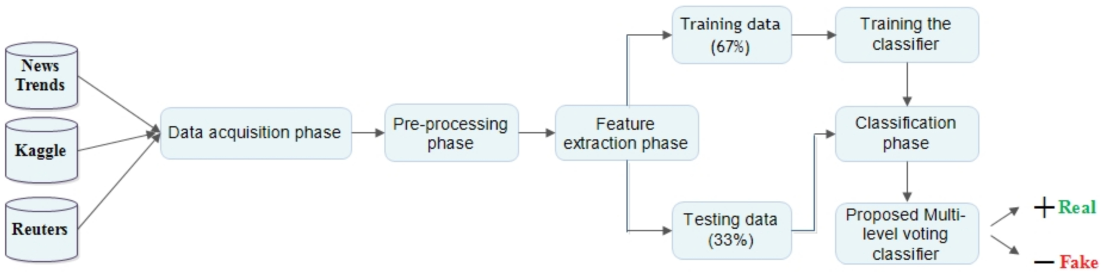


Fig. 1: Architecture of the proposed automatic fake news detection system

Overall statistics of our three collected datasets are discussed in Table 2. To analyze the length distribution of the titles for fake and real articles, Figure 2 is visualized. Where the X-axis labeled as 'title' represents the number of terms used in news article title's/headlines whereas Y-axis represents the corresponding number of articles having the same length distribution. A conclusion can be drawn from the mean distribution by visualising Figure 2 that the length of the title's or headlines of fake news articles is often longer than real news articles.

To further analyze the length distribution of the body content in articles, Figure 3 is observed, where the X-axis labeled as 'text' represents the number of terms used in news article texts whereas Y-axis represents the corresponding number of articles having the same length distribution.

It has been analyzed that the length of body/text of real news articles are often longer than fake news articles as shown in Figure 3(a) and 3(b) but for Reuters corpus the length of real news articles is more than fake news articles as seen in Figure 3(c).

The statistics of the mean distribution for Figure 2 and Figure 3 are compared in Table 3. In general, a conclusion can be drawn after analysing different datasets that the headlines of fake articles are longer in length and have shorter body content than real articles published on social networking [42] sites.

4.2 Pre-processing

In the pre-processing phase, non-semantic words such as prepositions, conjunctions, pronouns, *etc.*, also known as stop words are removed from the textual document as they provide very little or no information about fake content of an article. The redundant data in form of textual strings is removed from the document using a regular expression *(regex)* in the next step as shown in Figure 4. The *regex* and *pandas* library has been used to perform the pre-processing task [43]. *Regex* library has been used in python to define a search pattern using a sequence of characters whereas *dropna* method from *pandas* is used for cleaning the missing values in python DataFrame. We have used *random_state* function to select the entries from the dataset which is further used to split training and testing data points as it is used to split the data randomly.

To avoid overfitting, three standard splits (70:30, 67:33, 60:40) were used to perform the experiment. When the first standard split (70:30) was performed, it was observed that the data point dealt with an issue of underfitting. During the second split (60:40), overfitting of data was analyzed. Whereas, the third split (67:33) gave the best predicted line covering the majority of data points in the graph, so a standard split of 67:33 was chosen. The training data is then fed to feature generation phase as discussed in the next section.

4.3 Feature Generation

In order to extract numerical features from a textual document, tokenization, counting and normalization is done. During tokenization, each word is given a unique integer *ID*, following which occurrence of tokens is counted and then normalization of such tokens takes place. The whole process of converting the textual document into numerical feature vector is called as vectorization. Together this strategy (tokenization, counting, normalization) is called as 'Bag of n-grams' or 'Bag of Words' where n represents the continuous sequence of terms [44]. The three feature extraction techniques as shown in Figure 5 for retrieving features from the textual content of an article are Count-Vectorizer, TF-IDF and Hashing-Vectorizer which use *CountVectorizer*, *TfidfVectorizer* and *HashingVectorizer* classes from *feature_extraction* library of python, respectively.

4.3.1 Count Vectorizer (CV)

Count vectorizer is absolutely based on count of the word occurrences in the document. In count vectorizer technique, both counting the occurrences of tokens and

Table 2: Statistics of collected corpora

Corpus	Total article	Cleaned articles	Real articles	Fake articles	Published year
News Trends	7795	6335	3171	3164	2017
Kaggle	4048	3983	1865	2118	2017
Reuters	21578	19969	9622	10347	2004

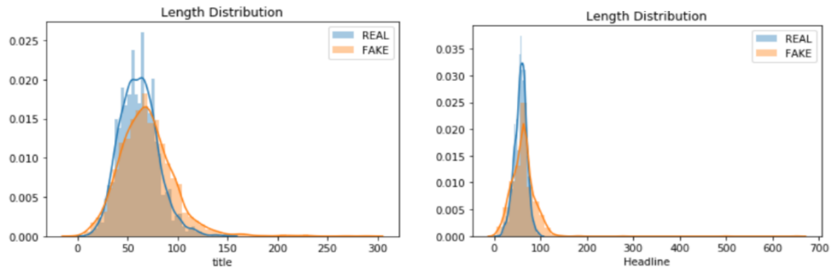


Fig. 2: Length distribution of headline for fake and real articles on (a) News Trends (b) Kaggle (c) Reuters corpora

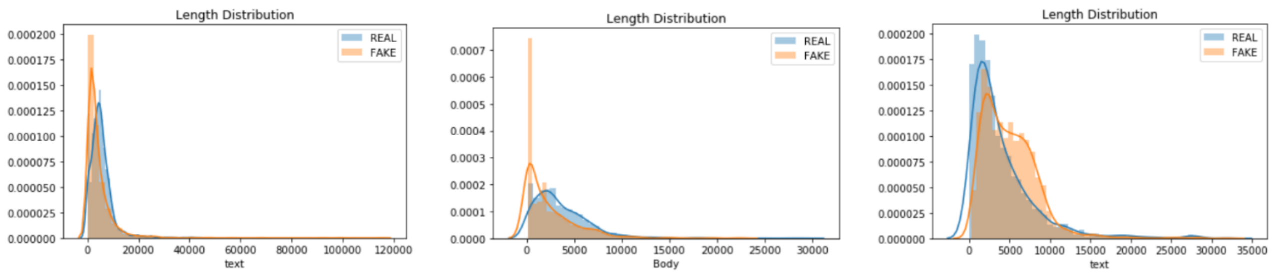


Fig. 3: Length distribution of text for fake and real articles on (a) News Trends (b) Kaggle (c) Reuters corpora

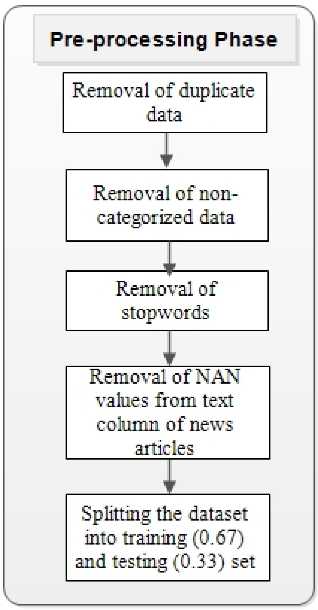


Fig. 4: Steps performed during the pre-processing phase

tokenization process is performed. Count vectorizer has many parameters for refining the type of features. One can build features using any of the three parameters, unigram *(min_df=1)*, bigram *(min_df=2)* and trigrams *(min_df=3)*. We have used *min_df* as 1 for our experiment. Here, each vector (term) in a document represents the individual feature name and its occurrence is depicted through a matrix to make it easier to understand as shown in Table 4. It has been observed from the above table, after pre-processing phase, the terms retrieved from the documents are represented as vectors on top of the sparse matrix and the frequency of terms in particular document is represented through count occurrences. Tag clouds of top 30 features retrieved after executing CV method is shown in Figure 6. It was observed that words like corruption, attacking, islamic, obama, losing, com are seen under fake tag clouds.

CV also counts the number of words occurring more frequently in the document, which may overshadow the words which occur less frequently but may have more importance to the document feature. This limitation of

Table 3: Mean distribution of labeled articles for News Trends, Kaggle and Reuters corpora

Corpus	**Mean distribution of title labeled as fake**	**Mean distribution of title labeled as real**	**Mean distribution of text labeled as fake**	**Mean distribution of text labeled as real**
News Trends	69.18	61.38	4121.04	5292.16
Kaggle	62.47	57.32	2380.82	3544.84
Reuters	81.79	71.29	5156.08	4540.26

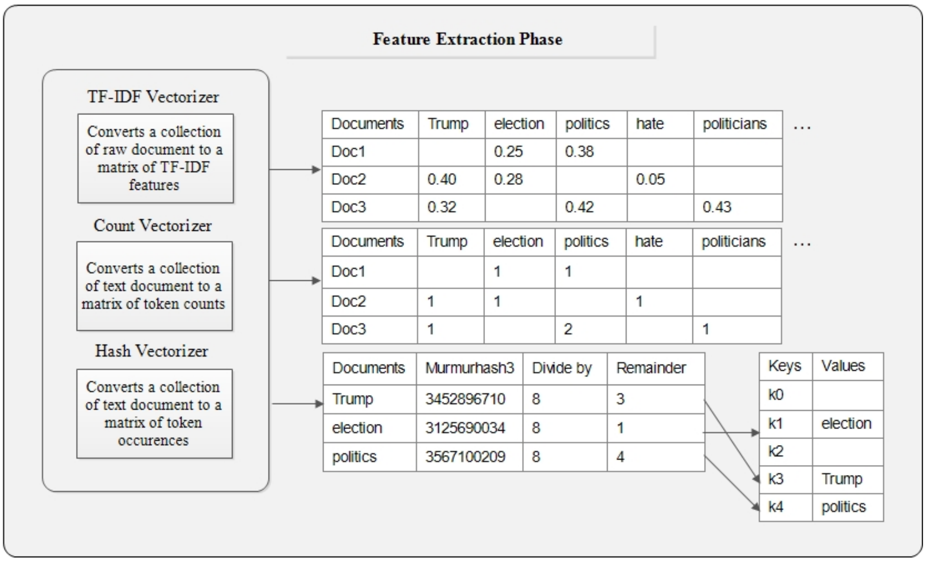


Fig. 5: Conversion of textual content into numerical vectors through TF-IDF, Count-vectorizer and Hash-vectorizer feature extraction techniques

Table 4: Sparse matrix representation using a Count-Vectorizer feature extraction technique

Document	Narendra	elections	vote	politics	Punjab	BJP	candidate
Doc1	0	1	2	0	0	2	1
Doc2	4	0	1	0	0	1	0
Doc3	1	3	2	0	1	0	1

CV can be handled by using TF-IDF feature extraction technique as explained below.

4.3.2 Term frequency- Inverse Document frequency (Tf-Idf)

Tf-Idf is a weighing matrix which is used to measure the importance of a term (count + weight) to a document in a dataset. The tokens retrieved from the textual data using both Tf-Idf and CV techniques are same, but weights assigned to the tokens of both techniques are different. Tf-Idf is composed of two metrics, named as, term frequency *(tf)* and inverse document frequency *(idf)*. Tf-Idf is represented by Eq. (1).

$$tfidf = tf(t,d) \times idf(t,d) \tag{1}$$

Here, Term Frequency denoted as *tf* and is calculated from the count *(c)*, term *(t)* in document *(d)* and represented as $tf\ (t,d) = c_{td}$. The frequency of occurrence of words to a binary feature is converted by using *1* (present in document) and *0* (not present in document). The frequencies can be normalized using average and logarithms. The inverse document frequency *(idf)* for a word w in document text *(t)* as computed by Eq. (2).

$$idf(t,d) = 1 + log\frac{T}{(1 + df(t))} \tag{2}$$

Here, T represents the total document count in our corpus and *df (t)* represents the count of the number of documents where the term t is present. The product of two measures will help to compute *tfidf*. Euclidean's normalized form is used to calculate the final Tf-Idf

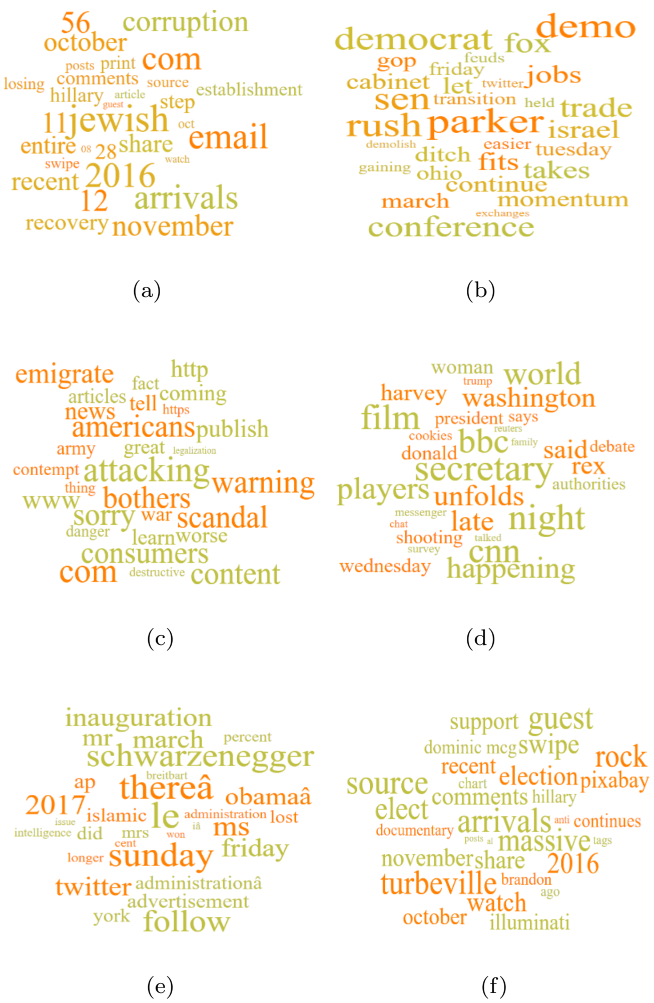


Fig. 6: Top 30 (a) News Trends fake, (b) News Trends real, (c) Kaggle fake, (d) Kaggle real, (e) Reuters fake, (f) Reuters real word clouds using count-vectorizer feature extraction technique

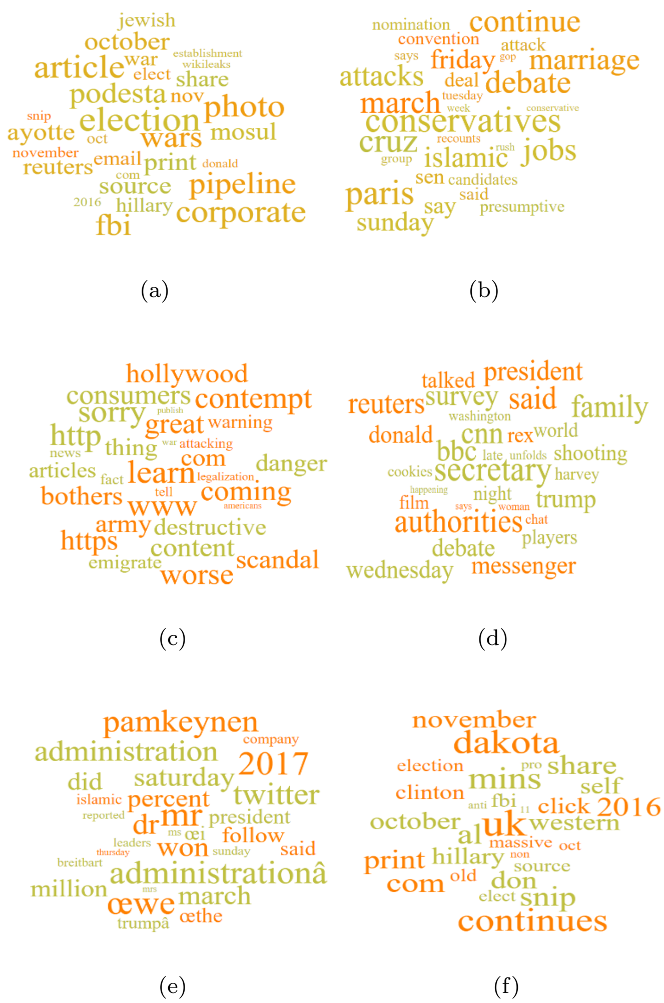


Fig. 7: Top 30 (a) News Trends fake, (b) News Trends real, (c) Kaggle fake, (d) Kaggle real, (e) Reuters fake, (f) Reuters real word clouds using TF-IDF feature extraction technique

metric as given by Eq. (3).

$$tfidf = \frac{tfidf}{||tfidf||} \quad (3)$$

Here $||tfidf||$ is the Euclidean norm. Tag clouds of top 30 features for fake and real articles using TF-IDF feature extracting technique is shown in Figure 7. It was observed that words like *www, danger, sorry, wars, islamic* are most common under fake news articles. The difference between count-vectorizer and TF-IDF approach is that the tokens retrieved from textual data are same but both have different weights assigned to each token being extracted.

4.3.3 Hashing Vectorizer (HV)

It is a memory-efficient technique. Unlike previous two techniques, tokens are stored as a string and here hashing trick is applied to encode the features as numerical indexes. Let us discuss the concept of HV using an example shown in Figure 8. When data is entered, the hashed attributes of data are retrieved. The hashed terms like Trump, election and politics are extracted from the document. In the next step, the hashing trick is applied on the hashed attributes, where a *Murmurhash3* function is applied on the hashed terms to generate a random number. Further, the assigned random numbers are divided by 8 and are stored in different keys such as *k2, k3, k4* based on the remainders retrieved after applying the *murmurhash3* function which is used for hash-based lookup. There is a possibility of collision

Documents	Murmurhash3	Divide by	Remainder
Triump	3452896710	8	3
election	3125690034	8	2
politics	3567100209	8	4

Keys	Values
k0	-
k1	-
k2	election
k3	Triump
k4	politics

Fig. 8: Hashing trick on textual data using Murmurhash3 function to get values in a specific range

when data have equal hashed attributes.

Let us suppose in our example document, we have Trump and politics words as an important keys which are seen more than once, thus causing collisions at *k3, k4* positions. The collided values are then occupied by other vacant positions in a set of documents. Such collision processing is dealt with parallel processing. The process is conceptually explained in Figure 9. These six

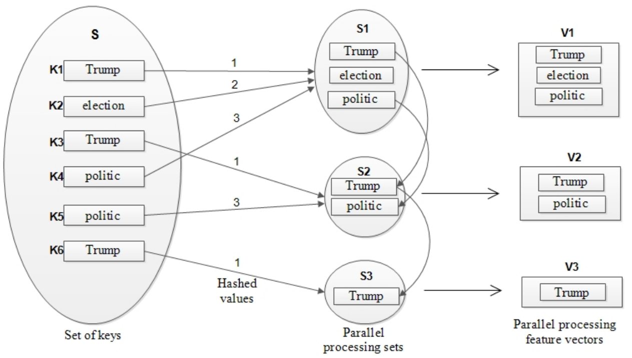


Fig. 9: Entry of redundant data into a hash table during parallel processing

terms are assigned six keys as shown in Figure 9 and are entered in the hash table. The hash values of keys *k1, k3, k6* are same, *i.e. Trump*; *k4* and *k5* are same *i.e.* politics, but rest have different values. Due to collision *k1, k3* and *k6* cannot be placed in same set *(S1)*. To enable parallel processing, *k1, k3 and k6* are placed in different sets. No two same hash values can be placed in a single set. Different keys like *k1, k2* and *k4* can be placed in the same set *(S1)* as these keys have different values. Values at keys, *k3* and *k6* are same so cannot be processed parallelly, therefore are processed in different sets. The values in *S1, S2, S3* are organized into vectors (numerical features) and can be processed using vector operations.

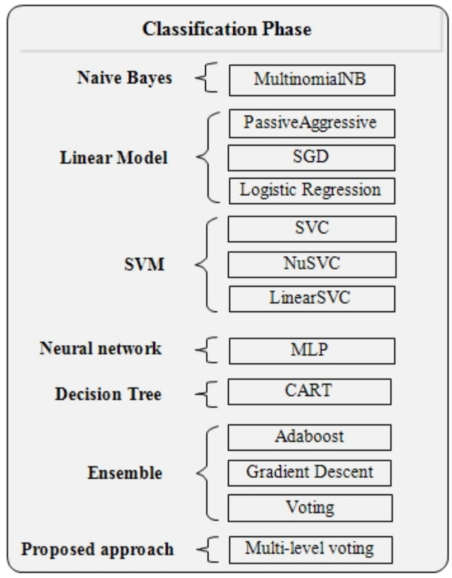


Fig. 10: Classification phase

The drawback of HV is that there is no way to get the feature names from feature indices, *i.e.*, the inverse transform cannot be used to compute the most important feature names through Hash-Vectorizer, unlike rest two methods.

5 Classification Algorithms

The processed dataset retrieved after pre-processing and feature extraction phase is then fed to the classification phase for the identification of fake news article. In this paper, six machine learning techniques, *i.e.*, Naïve Bayes (MultinomialNB), Support Vector Machine (Support Vector Classifier (SVC), NuSVC, LinearSVC), Decision Tree (CART), Linear (Passive Aggressive (PA), Stochastic Gradient Descent (SGD), Logistic Regression (LR)), Neural Network (Multi-Layer Perceptron (MLP)) and Ensemble models (AdaBoost, Gradient Boosting, Voting) have been applied as shown in Figure 10.

Classifier functions help to map the input feature vectors $f \varepsilon F$ to output labels $l \varepsilon \{1, 2, 3, \ldots, n\}$, where F is the feature space. The feature space is represented as, $F = \{Fake, Real\}^R$, where R is the real number. Our aim is to learn the classifier function from a labeled training data.

5.1 Naïve Bayes (NB)

It is a type of probability classifier. It works on Bayes Theorem and handles both categorical and continuous variables. NB assumes that every pair of features with labeled value is independent of each other. Given a collection of D documents from news articles, $D_i = \{d_1, d_2, ..., d_n\}$, where each document consists of T terms such as $D_i = \{t_1, t_2, ..., t_m\}$. Then the probability of D_i occurrence in class label C_l is given by Eq. (4).

$$P(C_l|D_i) = P(C_l) \prod_{n=1}^{m} P(d_n|C_l) \tag{4}$$

Here, the conditional probability of term t_m present in a document of class label C_l and the prior probability of document occurring in class label C_l is denoted by *P(C_l)*.

Multinomial Naïve Bayes (MutinomialNB) is a type of Naïve Bayes algorithm used for text classification. Data used in text classification to apply MultinomialNB can be represented as *TF-IDF* vectors, hashing vectors and count vectors. The feature vectors $V_f = (V_{f1}, V_{f2}, ..., V_{fn})$ are parameterized for each class C_n in the distribution, where n represents the feature numbers. The likelihood of observing V_f is given by Eq. (5).

$$P(C_n|V_f) = \frac{X_i!}{\prod_{n=1}^{p}} f_n! \prod_{n=1}^{P} V_n f_n \tag{5}$$

Here, f_n is the number of times the n^{th} feature has occurred in the document, X_i is the number of draws taken from the bag of features. $V_n^{f_n}$ and $f_n!$ are computed from training data.

5.2 Linear Model

The linear model helps to classify the group by making linear combinations of feature vectors. Linear classifiers work well with many features but it works best for document (features are extracted from the text) classification. If $\mathbf{v}$ is the input feature vector to the classifier, then the resultant score is given by Eq. (6).

$$s = f(\mathbf{vw}) = f(\sum_i w_i v_i) \tag{6}$$

here $\mathbf{w}$ is the weight of a feature vector and the function f gives the desires output of two vectors. The three linear models used in this paper are Passive Aggressive (PA), Stochastic Gradient Descent (SGD) and Logistic Regression (LR) classifiers. The PA algorithm has similar behavior with perceptron classifier in terms of learning rate but has dissimilar behaviour in terms of regularization parameter. The PA classifier is equivalent to PA-l [34] when the loss parameter is *hinge* and PA-ll [34] when the loss parameter *squared_hinge*. Second linear model used in this paper is SGD. The SGD model is updated with the decreasing learning rate after each sample interval and by default the loss parameter used in this paper is *hinge*. The classifier also allows minibatch learning. The LR model can be used for both multi and binary problem classification. Any other input format apart from *float64* gets converted in this classifier. All three linear models discussed above can take both sparse and dense matrix as their input.

5.3 Support Vector Machine (SVM)

SVM works on Structural Risk Minimization Principle. It is defined by a 'best' separating hyperplane and is also called as a discriminative classifier. Through the SVM model, feature vectors retrieved from text document of news articles are represented as points in feature space. Then the feature vectors are mapped in such a way that a wide gap is visible to perform linear classification. In our dataset the feature vectors are marked by making two categories, $C = \{Fake,\ Real\}$, then the training classifier builds a model which assigns new feature vectors to both defined categories.

SVM classes such as Linear Support Vector Classifier (LinearSVC), Nu-Support Vector Classifier (NuSVC), Support Vector Classifier (SVC) are used for performing classification on the dataset. NuSVC and SVC are almost similar but use slightly different sets of parameters and their mathematical formulations also vary. Whereas LinearSVC is another type of Support Vector Classification (SVC) and uses the case of linear kernel. All three classes take input in the form of two arrays with array X having 2-dimensional size [sample_number, feature_vectors] to handle training data and array Y having 2-dimensional size [category_label, sample_number]. The Decision function is the same for both SVC and NuSVC is given by Eq. (7).

$$sgn(\sum_{f=1}^{n} y_f \alpha_f K(V_f, V) + \mu) \tag{7}$$

where V_f are the training feature vectors, $f = 1, 2, ..., n$ in two categories. $K(V_f, V)$ is the kernel and $y_f \alpha_f$ is the dual coefficient parameter which holds support vectors and an independent intercept term μ. The only difference between SVC *($C=[0,\infty]$)* and NuSVC *($C=[0,1]$)* is seen from parameter C which is the penalty parameter of the error term. The class LinearSVC supports both sparse and dense input and is implemented in terms of liblinear so is more flexible in terms of the loss function and penalties, and scales better to large testing samples.

5.4 Neural Network (NN)

Neural networks are composed of highly interconnected neurons to solve specific problem parallelly. In this paper, Multi-Layer Perceptron (MLP) is implemented on our collected dataset. The classifier can be trained on either regression or classification dataset. The feature vectors $V_f = \{v_1, v_2, \ldots, v_n\}$ are retrieved after feature extraction phase and through training dataset the classifier learns a given function in Eq. (8).

$$f(n) : Ri \rightarrow Ro \tag{8}$$

where i is the dimensions for input and o is the dimensions for an output. In MLP there can be one or more non-linear layer (hidden layer) between the input and output layer. The input layer made up of neurons, where each neuron represents the input feature which is fed into the hidden layer. Then the hidden layer computes the weighted summation $w_1v_1 + w_2v_2 + \ldots + w_iv_i$, followed by function $f(n)$. The output value is given by the last hidden layer and is received by the output layer.

5.5 Decision Tree (DT)

The DT classifiers can be used for both regression and classification. The classifier predicts the target variable by learning the feature data and dividing the area into sub-regions. Based on two criteria multiple features are divided, one is a measure of impurity and other is information gain. In our dataset *'gini'* is the chosen impurity measure to calculate the highest information gain at each node for dividing the DT. In case of document data, the conditions depend upon the particular term in a text document of the news article. The data is divided repetitively until the leaf node cannot be further divided acquiring least information on them. The majority count of labels in the leaf node are used for classifying the textual data.

5.6 Ensemble methods

Such methods help to build a learning algorithm by combining the estimators to get robust classifier over single classifier. The boosting methods implemented on our dataset are Gradient Descent Boosting (GDB) and AdaBoost classifier for binary classification. The AdaBoost classifier assigns more weight to feature vectors which is difficult to handle and less weight to the features which can be easily handled. This process is repeated until the classifier correctly classifies the training data. The GDB model works on three elements such as weak learner, loss function and additive model as discussed in [16].

The other type of classifier that can be useful for balancing individual weakness of ML models is Voting classifier. In this paper, the Voting classifier has predicted the categories based on 'hard' voting which classifies the sample based on majority class label. To evaluate the classifiers discussed above, the performance metrics are defined and corresponding experimental results are discussed in the next section.

6 Evaluation Phase

The performance measures for binary classifiers applied in this paper has been evaluated with the help of a confusion matrix defined by four cells as shown in Table 5, where:

- cell 'a' counts the predicted document as 'Fake' when actually it is 'Fake', known as true positive (TP) rate.
- cell 'b' counts the predicted document as 'Real' when actually it is 'Fake', known as false positive (FP) rate.
- cell 'c' counts the predicted document as 'Fake' when actually it is 'Real', known as false negative (FN) rate.
- cell 'd' counts the predicted document as 'Real' when actually it is 'Real', known as true negative (TN) rate.

Table 5: A confusion matrix representation

Actual↓ Predicted→	**Fake**	**Real**
Fake	TP (a)	FP (b)
Real	FN (c)	TN (d)

The conventional performance measure has been evaluated from the above confusion matrix cells. The measures computed from the matrix are precision as represented by Eq. (9), recall by Eq. (10), specificity by Eq. (11), accuracy by Eq. (12), error by Eq. (13), F1-score by Eq. (14) as shown below.

$$Precision\ (Pr) = \frac{a}{a+b} \tag{9}$$

$$Recall\ (Re) = \frac{a}{a+c} \tag{10}$$

$$Specificity\ (Sp) = \frac{d}{d+b} \tag{11}$$

$$Accuracy\ (Acc) = \frac{a+d}{n}, where\ n = a+b+c+d > 0$$

(12)

$$Error\ (Err) = \frac{b+c}{n}, where\ n = a+b+c+d > 0 \quad (13)$$

$$F1 - score\ (F1) = \frac{2 \times Pr \times Re}{Re + Pr} \quad (14)$$

The predictions made by the classification models are evaluated in this phase based on their performance metrics. The most intuitive performance measure is accuracy, which helps to predict the best model. Various machine learning models used in experiment are MultinomialNB ($C1$), Passive Aggressive ($C2$), Stochastic Gradient Descent ($C3$), Logistic Regression ($C4$), SVC ($C5$), NuSVC ($C6$), LinearSVC ($C7$), Multi-Layer Perceptron ($C8$), Decision Tree ($C9$), AdaBoost ($C10$), Gradient Descent ($C11$), Voting ($C12$) and Multi-level Voting ($C13$) classifiers. To evaluate these models, a comparative analysis has been shown in Figure 11. The experiment is performed on three (News Trends, Kaggle, Reuters) different corpus. In this paper, Tf-Idf, CV, and HV feature extraction techniques are used to extract the feature vectors from the documents of the chosen corpus. In Table 6, the accuracy measure of various ML classifiers is compared.

The best accuracy retrieved by top 3 models in all three corpora is Linear Support Vector Classifier (LSVC), Passive Aggressive (PA) and Logistic Regression (LR) classifiers.

Other parameters used to evaluate the performance measure of classifiers used in this paper are precision, recall and F1-score. The precision metric helps to calculate the proportion of news article that are predicted fake and actually also belongs to the fake news article category. The comparative analysis of precision metric is shown in Table 7.

The recall metric helps to calculate the proportion of news article that are predicted fake but actually belongs to both fake and real articles. The comparative analysis of recall metric is shown in Table 8. The specificity metric helps to calculate the proportion of news article that are correctly predicted as a real news article known not to be fake. Specificity is measured as inverse of recall metric. The comparative analysis of specificity metric is shown in Table 9. Accuracy is only measured as a strong metric when both false negative and false positive values are closer to each other else the metric is not considered as a good performance measure. To convey a balance between recall and precision, F1-score performance metric is selected to retrieve the best model. F1-score metric helps to take both false positives and false negatives into account. The comparative analysis of F1-score metric is shown in Table 10.

The objective of ROC is to notice the increase in false positive rates (FPR) with an increase in true positive rates (TPR) with a varying threshold of the classifiers used in this paper. The performance of class models at various thresholds is shown through graphs in Figure 12. The curve drawn in the graph is known as Receiver Operating Characteristic (ROC) curve. The ROC for News trends, Kaggle and Reuters are plotted using two parameters such as FPR and TPR as given by Eq. (15) and Eq. (16).

$$TPR = \frac{a}{a+c} \quad (15)$$

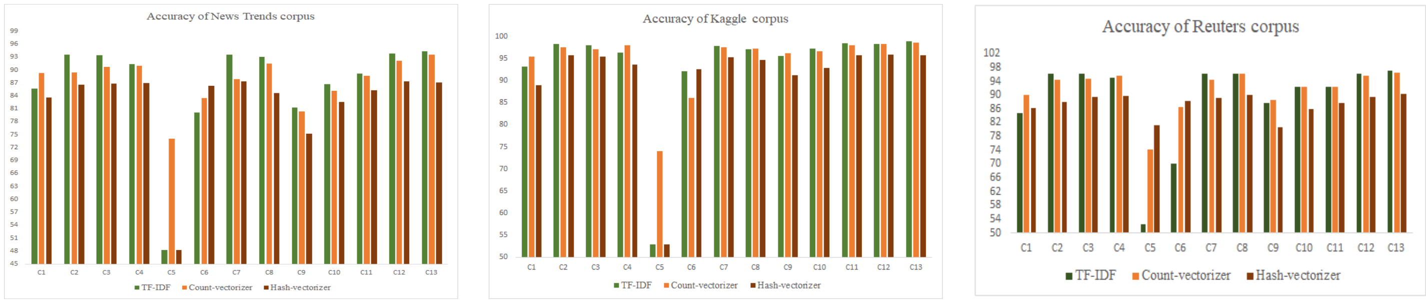
$$FPR = \frac{b}{b+d} \quad (16)$$

Here, a, b, c, and d represents the TP, FP, FN and TN rates, respectively. predicted document as 'Real' when actually it is 'Fake', known as false positive (FP) rate, c counts the predicted document as 'Fake' when actually it is 'Real', known as false negative (FN) rate, d counts the predicted document as 'Real' when actually it is 'Real', known as true negative (TN) rate.

6.1 Major findings

The major findings deal with the question of whether a trained classifier using old news articles can give accurate and efficient results to categorize the differences between fake and real content. It has been observed that the performance of classifying news article depends on the corpus and type of classification model. In our experiment, three corpora have been collected from three different sources [39] [40] [41]. Each corpus is divided into training (0.67) and testing (0.33) sets. The experiment was performed on these chosen datasets using Term frequency-Inverse document frequency (Tf-Idf), Count-Vectorizer (CV) and Hashing-Vectorizer (HV) feature extraction techniques. From the accuracy perspective, Passive Aggressive (93.2%) and LinearSVC (93.2%) outperform other models for all three (News Trends, Kaggle, Reuters) corpora. Whereas the Passive Aggressive (96%) and the LinearSVC (95.9%) performs best using Tf-Idf, Logistic Regression (94.9%) and Stochastic Gradient Descent (94.2%) performs best using CV, LinearSVC (90.6%) and Stochastic Gradient Descent (90.5%) performs best using HV individually for all three corpora.

A classifier is considered usable only if it achieves both high precision and recall. To average out the results of both precision and recall, F1-score is taken into consideration. On evaluating F1-score metric, it was observed that Passive Aggressive (93.3%), Stochastic



(a) (b) (c)

Fig. 11: Performance analysis using accuracy metric for (a) News Trends, (b) Kaggle and (c) Reuters dataset on basis of Tf-Idf, Hashing-Vectorizer and Count-Vectorizer feature extraction techniques

Table 6: Comparative analysis of accuracy measure using Machine Learning classifiers

MODELS	News Trends			Kaggle			Reuters		
	Tf-Idf	CV	HV	Tf-Idf	CV	HV	Tf-Idf	CV	HV
Multinomial Naïve Bayes	85.7	89.3	83.6	93.2	95.4	89	84.8	89.9	86.2
Passive Aggressive	**93.5**	89.4	86.6	**98.3**	**97.6**	**95.7**	**96.2**	94.3	88
Stochastic Gradient Descent	93.4	**90.7**	86.8	98	97.1	95.5	**96.2**	94.8	89.4
Logistic Regression	91.4	**91.0**	**87.0**	96.4	98	93.6	94.9	**95.7**	**89.6**
Support Vector Classifier	48.2	74.1	48.2	52.9	74	52.9	52.5	74.3	81.3
NuSVC	80.1	83.5	**86.3**	92.2	86.1	92.6	70	86.6	88.3
LinearSVC	**93.6**	87.9	**87.3**	**97.9**	**97.6**	95.3	**96.3**	94.4	89.3
Multi-Layer Perceptron	93	91.5	84.7	97.1	97.3	94.7	96.2	96.2	90.1
Decision Tree	81.3	80.4	75.2	95.6	96.2	91.3	87.7	88.5	80.6
AdaBoost	86.7	85.1	82.6	97.3	96.6	92.9	92.5	92.4	85.9
Gradient Boosting	89.2	88.6	85.3	98.5	**98.0**	**95.7**	92.3	92.5	87.6
Voting Classifier	93.8	92.1	87.3	98.3	98.3	95.9	96.1	96.4	90.3

Table 7: Comparative analysis of precision metric using Machine Learning classifiers

MODELS	News Trends			Kaggle			Reuters		
	Tf-Idf	CV	HV	Tf-Idf	CV	HV	Tf-Idf	CV	HV
Multinomial Naïve Bayes	73.3	85.8	88.4	92.3	95.8	84.4	99.1	97.3	89.3
Passive Aggressive	**94.5**	90.0	87.6	**98.7**	**96.9**	**97.1**	**96.6**	93.6	**91.1**
Stochastic Gradient Descent	94.8	**89.4**	88.1	**98.7**	97.8	**95.8**	**96.4**	94.2	90.0
Logistic Regression	95.4	**94.0**	**89.8**	95.4	97.9	94.2	94.7	**95.2**	89.1
Support Vector Classifier	100	96.6	100	100	97.2	100	100	53.1	82.0
NuSVC	94.5	96.3	90.4	97.5	97.9	91.6	41.9	77.8	85.7
LinearSVC	**96.1**	88.8	**87.8**	**98.1**	**96.5**	96.2	**93.6**	94.1	89.4
Multi-Layer Perceptron	93.6	93.1	84.0	97.2	96.5	96.2	92.1	94.3	89.9
Decision Tree	80.2	81.9	75.0	97.2	96.6	92.8	87.1	88.3	80.1
AdaBoost	89.9	88.7	83.6	97.2	96.6	94.1	92	90.6	84.9
Gradient Boosting	91.9	92.6	89.0	98.4	**97.8**	**95.4**	90.7	90.4	89.9
Voting Classifier	95.8	94.2	89.2	98.8	97.8	96.8	96.8	95.0	89.9

Gradient Descent (93.5%) and LinearSVC (93%) outperform other models on all three news article corpora using Tf-Idf, CV and HV feature extraction techniques. The Passive Aggressive (95.9%), Stochastic Gradient

Table 8: Comparative analysis of recall metric using Machine Learning classifiers

MODELS	**News Trends**			**Kaggle**			**Reuters**		
	Tf-Idf	**CV**	**HV**	**Tf-Idf**	**CV**	**HV**	**Tf-Idf**	**CV**	**HV**
Multinomial Naïve Bayes	95.9	91.5	79.7	94.6	95.4	94	77.9	85.4	85.0
Passive Aggressive	**92.2**	88.1	85.0	**98.0**	**98.3**	**94.8**	**96.0**	95.3	86.7
Stochastic Gradient Descent	91.7	**90.9**	84.9	**97.5**	96.7	**95.6**	**96.2**	95.7	89.7
Logistic Regression	87.7	**88**	**84.2**	97.7	98.5	93.7	95.5	**96.5**	**90.7**
Support Vector Classifier	48.2	65.7	48.2	52.9	67.4	52.9	52.5	96.3	82.3
NuSVC	72.4	75.9	**82.6**	88.7	80.2	94.2	99.4	95.8	91.4
LinearSVC	**91.1**	86.4	**86**	**97.8**	**98.9**	94.9	**96.6**	95.1	90.1
Multi-Layer Perceptron	91.9	89.9	84.1	97.2	98.2	93.8	96.4	95.8	89.8
Decision Tree	80.7	78.3	73.9	94.5	96.1	90.9	89.2	89.6	82.3
AdaBoost	83.5	81.8	80.9	97.5	96.8	92.6	93.6	94.5	87.8
Gradient Boosting	86.4	85.0	82.0	98.7	98.4	96.3	94.2	95.1	89.9
Voting Classifier	91.6	89.7	85.1	98.0	98.9	95.4	95.0	96.6	89.8

Table 9: Comparative analysis of specificity metric using Machine Learning classifiers

MODELS	**News Trends**			**Kaggle**			**Reuters**		
	Tf-Idf	**CV**	**HV**	**Tf-Idf**	**CV**	**HV**	**Tf-Idf**	**CV**	**HV**
Multinomial Naïve Bayes	79.6	87.5	88	91.6	95.2	84.3	98.7	96.4	87.4
Passive Aggressive	**94.8**	90.5	88.2	**98.5**	**96.6**	**96.6**	**96.2**	93.1	89.6
Stochastic Gradient Descent	95	**90.4**	88.6	**98.5**	97.5	**95.3**	**96**	93.7	88.9
Logistic Regression	95.3	**94**	**89.9**	94.9	**97.4**	93.4	94.2	**94.8**	88.2
Support Vector Classifier	0	94.4	0	0	95.1	0	0	65.3	80.2
NuSVC	92.9	95.4	**90.2**	96.9	96.5	90.9	61.3	79.6	85.2
LinearSVC	**96.2**	89.3	**88.5**	**97.8**	**96.2**	95.7	**95.9**	93.5	88.4
Multi-Layer Perceptron	93.9	93.4	85.1	96.9	96.1	95.6	94.3	93.2	86.1
Decision Tree	81.7	82.4	76.4	96.8	96.2	91.7	86.1	87.3	78.6
AdaBoost	89.9	88.6	84.2	96.9	96.2	93.2	91.3	90.1	83.9
Gradient Boosting	92	92.5	88.9	98.2	**97.5**	**94.8**	90.1	89.9	85.2
Voting Classifier	95.9	94.3	89.5	98.6	97.6	96.3	96.4	94.6	88.8

Table 10: Comparative analysis of F1-score metric using Machine Learning classifiers

MODELS	**News Trends**			**Kaggle**			**Reuters**		
	Tf-Idf	**CV**	**HV**	**Tf-Idf**	**CV**	**HV**	**Tf-Idf**	**CV**	**HV**
Multinomial Naïve Bayes	86.9	89.4	83.6	93.4	95.5	88.9	87.2	90.9	87.0
Passive Aggressive	**93.4**	89.2	86.5	**98.3**	**97.5**	**95.9**	**96.2**	94.4	**88.8**
Stochastic Gradient Descent	93.3	**90.6**	86.7	**98.0**	97.2	95.6	**96.2**	94.9	**89.8**
Logistic Regression	91.3	**90.9**	**86.8**	96.5	**98.1**	93.9	95.0	95.8	**89.8**
Support Vector Classifier	65	77.4	65	69.1	79.6	69.1	68.8	68.4	82.1
NuSVC	81.3	84.5	86.2	92.8	88.1	92.8	58.9	85.8	88.4
LinearSVC	**93.5**	87.8	**87.2**	97.9	**97.3**	94.9	**95**	**94.5**	89.7
Multi-Layer Perceptron	92.8	91.6	84.5	97.2	97.3	94.9	93.2	94.1	88.7
Decision Tree	81.1	80.2	75.1	95.8	96.3	91.8	88.1	88.9	81.1
AdaBoost	86.5	85.0	82.5	97.3	96.6	93.3	92.7	92.5	86.3
Gradient Boosting	89.1	88.5	85.3	98.5	**98.0**	**95.8**	92.4	92.6	89.9
Voting Classifier	93.7	91.8	87.1	98.3	98.3	96	95.8	95.7	89.8

Descent (95.8%) and LinearSVC (95.4%) performs best using Tf-Idf, Logistic Regression (94.9%), Stochastic

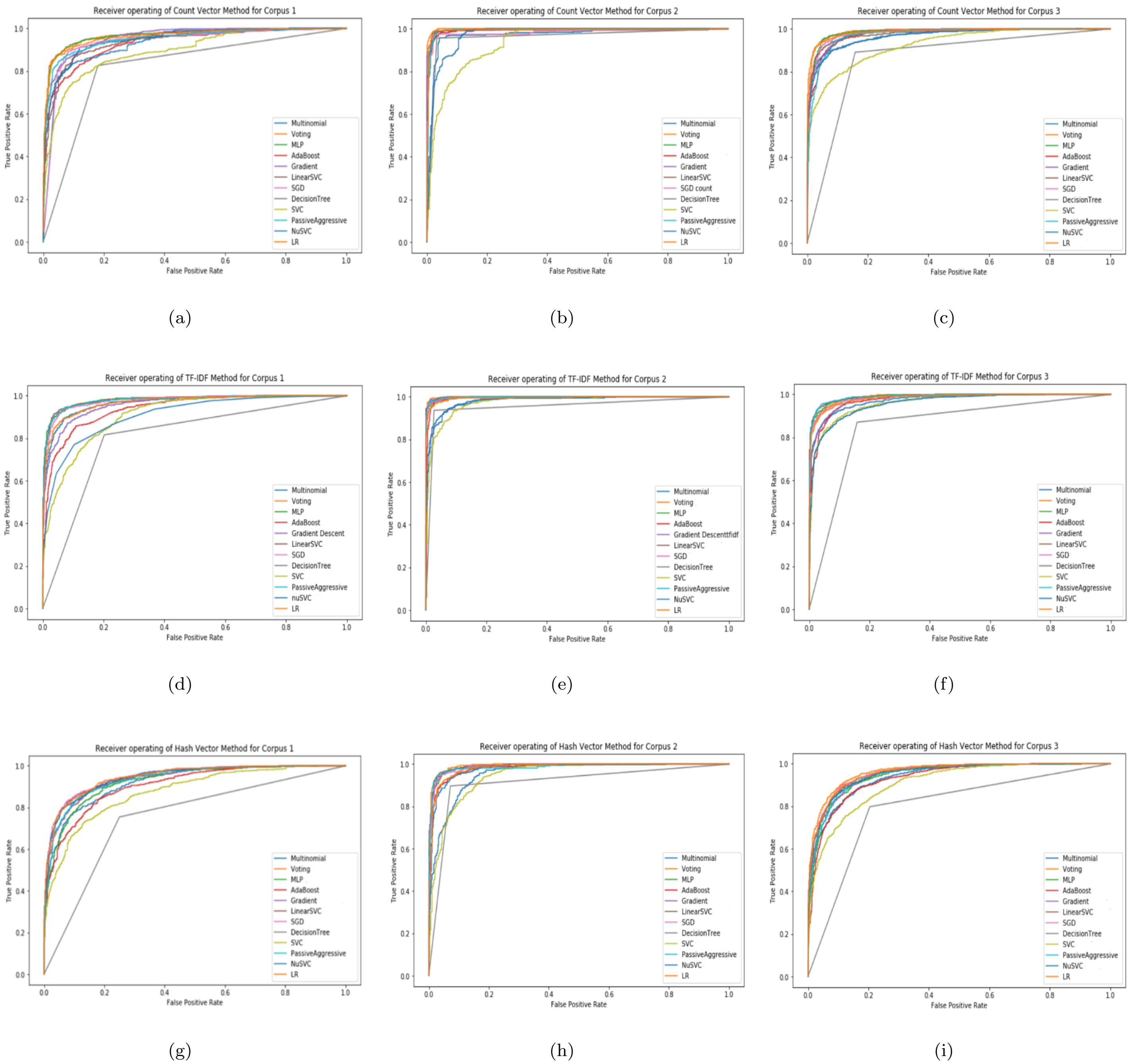


Fig. 12: TPR vs FPR at different classification thresholds for (a) Count-vectorizer on News Trends, (b) Count-vectorizer on Kaggle, (c) Count-vectorizer on Reuters, (d) TF-IDF on News Trends, (e) TF-IDF on Kaggle, (f) TF-IDF on Reuters, (g) Hash-vectorizer on News Trends, (h) Hash-vectorizer on Kaggle, (i) Hash-vectorizer on Reuters datasets

Gradient Descent (94.2%), Passive Aggressive (93.7%), LinearSVC(93.2) performs best using CV and Passive Aggressive (90.4%), Stochastic Gradient Descent (90.7%) and LinearSVC (90.6%) performs best using HV. The proposed multi-level voting model outperforms the Passive Aggressive model by 0.8%, 0.6%, 1.0% using Tf-Idf approach; outperforms the Logistic Regression by 2.6%, 0.7%, 0.8% using CV approach; outperforms the LinearSVC by 0.0%, 0.5%, 0.9% using HV approach on News Trends, Kaggle and Reuters corpus, respectively.

To evaluate the predictive performance of our approach, ROC_AUC is plotted. Passive Aggressive, Stochastic Gradient Descent and LinearSVC, Gradient Boosting, Logistic Regression outperform other ML models on basis of ROC_AUC metric with $TPR > 0.97$. Passive Aggressive, Stochastic Gradient Descent and LinearSVC gives $TPR > 0.97$ using Tf-Idf, Logistic Regression gives $TPR > 0.98$ using CV, Stochastic Gradient Descent, LinearSVC, Logistic Regression gives $TPR >$ 0.95 for News Trends, Kaggle and Reuters dataset, respectively. Based on the training time required by various ML classifiers, it has been observed that there is a trade off between efficiency and accuracy. The training time required by HV is less than Tf-Idf and CV technique but it compromises accuracy metric. It has been analyzed that the hashing technique is useful when the focus is to achieve high efficiency on a huge dataset. The two ML classifiers such as Logistic Regression and LinearSVC are chosen among other models which results in both high accuracy and efficiency to overcome the trade-off issue.

7 Proposed multi-level voting model

The proposed multi-level voting model not only helps to improve the accuracy but also helps to reduce the training time of the classifiers. The reduction in training time helps to increase the efficiency of our model by introducing parallel methods where the base learners are generated parallelLy. The motivation behind the proposed model is to analyze the independence between the base learners. Three levels are proposed to perform the experiment as discussed below.

Level 1: Sets of three ML classifiers are merged based on their performance metric (FP rate) to apply voting classifier. The voting models ($VC1, VC2, VC3$) are retrieved.

Level 2: A voting classifiers ($VC4$) is retrieved after merging the three models ($VC1, VC2, VC3$) on the basis of their false positive rate.

Level 3: The false predictions from voting classifier ($VC4$) are merged with PA and LinearSVC for Tf-Idf, LR and SGD for CV and SGD, LinearSVC for HV to get the final prediction.

On the basis of the minimum false positive (FP) rate, the ML models are merged to overcome the weakness of existing individual models. The minimum is the FP ratio, the more accurate the model will be to predict the content as fake. Three feature extraction techniques (Tf-Idf, CV, HV) are used to extract the features from a collected dataset. Based on the FP ratio, the models are selected and merged to give an appropriate prediction. First cluster at level 1, SGD (News Trends), LR (Kaggle) and MLP (Reuters) using Tf-Idf; SGD (News Trends), LinearSVC (Kaggle) and MLP (Reuters) using CV; PA (News Trends), LR (Kaggle) and MLP (Reuters) using HV are merged to build the voting classifier ($VC1$). Second cluster at level 1, AdaBoost, Gradient Boosting and MultinomialNB (News Trends, Kaggle and Reuters) using Tf-Idf, CV and HV are merged to build $VC2$. Third cluster at level 1, SVC, NuSVC and DT (News Trends, Kaggle and Reuters) using Tf-Idf, CV and HV are merged to build $VC3$ as shown in Figure 13.

Based on FP ratio of three proposed voting classifiers ($VC1, VC2, VC3$), a fourth voting classifier ($VC4$) is retrieved at level 2 based on the FP rates of $VC1, VC2$ and $VC3$ classifiers. At level 3, PA, LinearSVC using Tf-Idf; SGD and LR using CV; SGD and LinearSVC using HV are retrieved based on TP rate and are further clustered with $VC4$. Based on FP rate, $VC5$ is retrieved to give the final prediction of the proposed model.

8 Evaluating the performance of multi-level voting model

In proposed multi-level voting classifier, the best three ML models are combined from each feature extraction technique as discussed in the previous section. It can be observed from Table 11 that the proposed model outperforms voting classifier by 0.73%, 0.66% and 0.13% using Tf-Idf, CV and HV, respectively in terms of accuracy metric. Similarly, the proposed model also gives significant improvement for precision, recall, specificity, and F1-score performance measures.

To compare the vectorization methods used in terms of training time required to train the data, Figure 14 depicts that the HV technique is more efficient than other feature extraction methods as the time required to train the proposed multi-level voting model is least for News Trends and Reuters dataset. To achieve better efficiency (less training time), Tf-Idf method is only suitable when data is not too huge whereas hashing can be used for huge datasets when there is a requirement to make a trade-off between efficiency and accuracy. After performing our experiment, some conclusions are drawn in next section which helps to analyze the best classifier to be chosen for both high efficiency and accuracy.

9 Comparison with Existing Works

The results given by our system are also compared with other existing works on fake news detection as shown

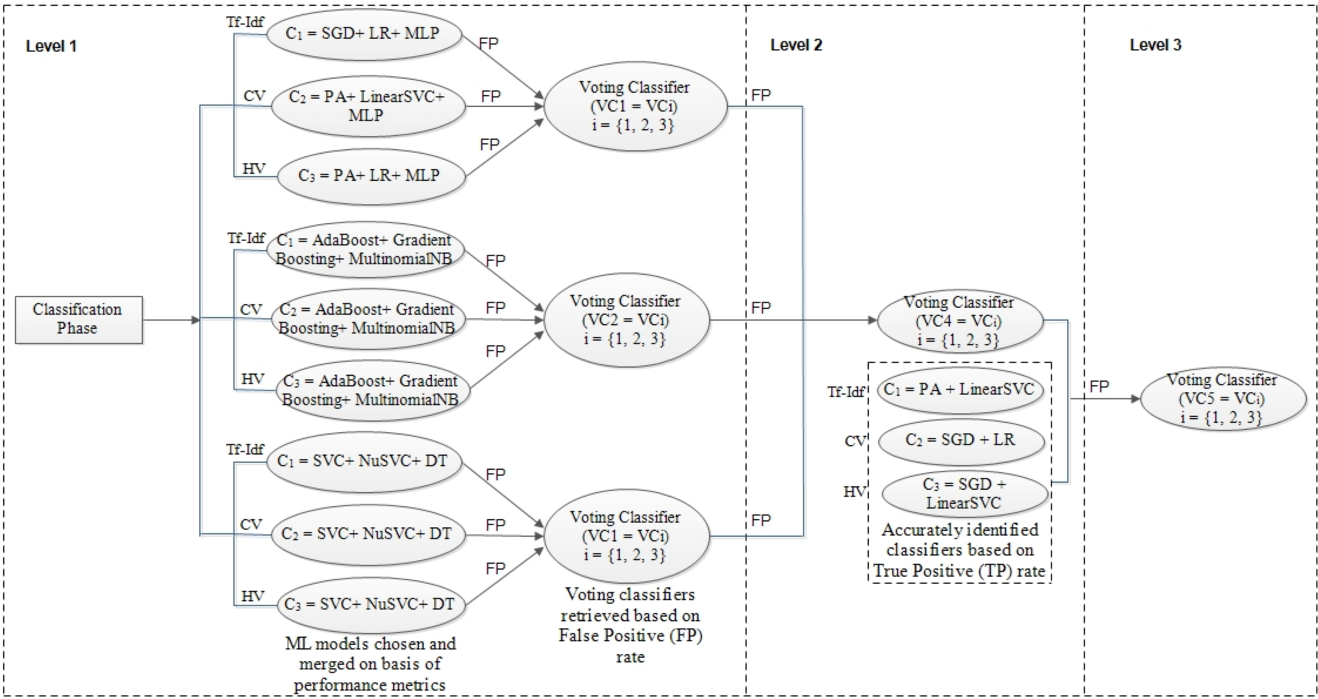


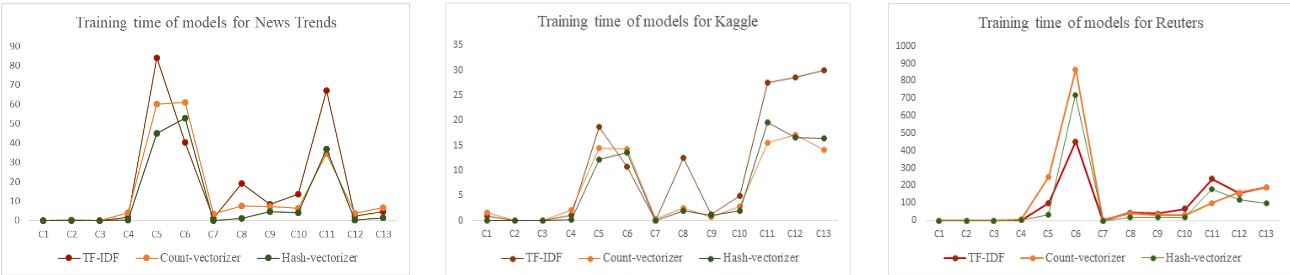
Fig. 13: Architecture of the proposed multi-level voting model

Table 11: Comparative analysis of the proposed multi-level voting model with voting classifier

MODELS	News Trends			Kaggle			Reuters		
	Tf-Idf	CV	HV	Tf-Idf	CV	HV	Tf-Idf	CV	HV
	Accuracy								
Voting classifier	93.8	92.1	87.3	98.3	98.3	95.9	96.1	96.4	90.3
Multi-voting classifier	94.3	93.6	87.1	98.9	98.7	95.8	97.2	96.5	90.2
	Precision								
Voting classifier	95.8	94.2	89.2	98.8	97.8	96.8	96.8	95	89.9
Multi-voting classifier	96.4	94.3	89.6	99.1	98.3	96.8	98.4	97.6	90.4
	Recall								
Voting classifier	91.6	89.7	85.1	98	98.9	95.4	95	96.6	89.8
Multi-voting classifier	93.1	91.4	85.2	98.7	98.8	94.4	96.8	95.2	90.8
	Specificity								
Voting classifier	95.9	94.3	89.5	98.6	97.6	96.3	96.4	94.6	88.8
Multi-voting classifier	96	92.4	86.1	98.2	97.1	93.6	95.7	94.6	87.9
	F1-Score								
Voting classifier	93.7	91.8	87.1	98.3	98.3	96	95.8	95.7	89.8
Multi-voting classifier	94.7	92.8	87.3	98.8	98.7	95.5	97.7	97	90.5

in Table 12, Table 13 and Table 14 using Reuters, Kaggle and News Trends corpora, respectively. Researchers have used ML approaches to perform fake news detection on various social media platforms. It has been analyzed from Table 12 that the feature extraction technique based on document frequency used by Mishu *et al.* (2016) for MultinomialNB (72%), SVC (78%) and voting classifier (89%) [45] does not give better accuracy than our proposed system for all three classifiers. In our proposed system, MultinomialNB (89.9%), SVC (81.3%), Voting (96.4%) outperforms Mishu *et al.* (2016) models.

It has been observed from Table 13 that the feature extraction technique based on Tf-Idf used by Ahmed *et al.* (2017) for ML classifiers does not gives better accuracy than our proposed system. The recorded observa-



(a) (b) (c)

Fig. 14: Training time comparison for (a) News Trends, (b) Kaggle and (c) Reuters dataset on basis of TF-IDF, Hash-vectorizer and Count-vectorizer feature extraction techniques

Table 12: Comparison of existing models with a proposed system for Reuters corpus

Authors	**Proposed Approach**	**Features**	**Model Accuracy**
Cai *et al.* (2008)	Presented a Bayesian classification approach using class-specific features for automatic text classification.	Topic-based and document-based modeling.	LapPLSI - 74.6%
Mishu *et al.* (2016)	Classified text document using various classifiers	Document frequency	MultinomialNB- 72% LR- 73.5% SGD- 76% SVC- 78% LinearSVC- 83.3% Voting- 89%
Analysis based on individual ML models	Classified articles as fake or real using various classifiers.	TF-IDF	SGD- 96.2% PA- 96.2% LinearSVC- 96.3% MLP- 96.2% AdaBoost- 92.5% Voting- 96.4%
		Count-vectorizer	MLP- 96.2% DT- 88.5% LR- 95.7% MultinomailNB-89.9% Gradient Boosting- 92.5% SVC- 81.3%
		Hashing-vectorizer	NuSVC- 88.3%
Proposed Multi-level voting model	Fake news detection system is proposed to achieve high accuracy and high efficiency.	TF-IDF	Multi-level voting- 97.2%
		Count-vectorizer	Multi-level voting- 96.5%
		Hashing-vectorizer	Multi-level voting- 90.2%

tion shows that our proposed multi-level voting model (97.2%) outperforms Gradient Boosting by 0.4% using Tf-Idf, LR by 2.3% using CV and NuSVC by 3.6% using HV feature extraction techniques, respectively for Kaggle corpus. From Table 14, it can be analyzed that our proposed multi-level voting model outperforms PA by 0.8% using Tf-Idf, MultinomialNB by 4.3% using CV and NuSVC by 0.8% using HV feature extraction techniques, respectively for News Trends corpus. From the comparative analysis, it has been analyzed that our proposed multi-level voting model using all three feature extraction technique outperforms when compared to the individual performance metrics of ML models used by Ahmed *et al.* (2017) and Mishu *et al.* (2016)

Table 13: Comparison of existing models with a proposed system for Kaggle corpus

Authors	**Proposed Approach**	**Features**	**Model Accuracy**
Ahmed *et al.* (2017)	Proposed a fake news detection model using n-gram analysis and ML techniques	n-gram based TF and TF-IDF	LSVM-92% KNN- 83.1% SVM- 86% DT- 89% SGD- 89% LR- 89%
Analysis based on individual ML models	Classified articles as fake or real using various classifiers.	Count-vectorizer	MultinomailNB- 93.2% SVC- 52.9% LR- 96.4% MLP- 97.1% DT- 95.6% Voting- 98.3%
		TF-IDF	LinearSVC- 97.9% SGD- 98% PA- 98.3% AdaBoost- 97.3% Gradient Boosting- 98.5% Voting- 98.3%
		Hashing-vectorizer	NuSVC- 92.2%
Proposed Multi-level voting model	Fake news detection system is proposed to achieve high accuracy and high efficiency.	TF-IDF	Multi-level voting- 98.9%
		Count-vectorizer	Multi-level voting- 98.7%
		Hashing-vectorizer	Multi-level voting- 95.8%

Table 14: Comparison of existing models with a proposed system for News Trends corpus

Authors	**Proposed Approach**	**Features**	**Model Accuracy**
Kuleshov *et al.* (2018)	Author shows the existence of adversarial examples in natural language classification	n-gram	NB- 93%
Analysis based on individual ML models	Classified articles as fake or real using various classifiers.	TF-IDF	PA- 93.5% SGD- 93.4% LinearSVC- 93.6% MLP- 93% DT- 81.3% AdaBoost- 86.7% Gradient Boosting- 89.2% Voting- 93.8% LR- 91.4%
		Count-vectorizer	MultinomailNB-89.3% SVC- 74.1%
		Hash-vectorizer	NuSVC-86.3%
Proposed Multi-level voting model	Fake news detection system is proposed to achieve high accuracy and high efficiency.	TF-IDF	Multi-level voting- 94.3%
		Count-vectorizer	Multi-level voting-93.6%
		Hash-vectorizer	Multi-level voting- 87.1%

for developing a system for automatic classification of news article features to label them as fake or real news article. It has been observed from Table 12 that the proposed model outplays the PA model by 0.9% using Tf-Idf, LR model by 0.8% using CV and NuSVC model by 1.9% using HV feature extraction techniques, respectively when compared with their performance metrics.

10 Conclusion and Future Scope

The goal in this paper is to analyze the best known supervised technique for detecting fake news. When dealing with the machine learning classifiers, the key question is not to find a learning classifier superior to others but rather to find the conditions under which particular model outperform others for a given problem. The set of attributes extracted from the corpus taken uses three feature extraction techniques (Tf-Idf, CV and HV) to feed the extracted feature vectors into the selected machine learning models. Some characteristics taken from datasets for learning task are categorical attributes, missing values, headlines of the article, the body of article and the publisher name. Various classifiers, Multinomial Naïve Bayes (MultinomialNB), Passive Aggressive (PA), Stochastic Gradient Descent (SGD), Logistic Regression (LR), Support Vector classifier (SVC), NuSVC, LinearSVC, Multi-Layer Perceptron (MLP), Decision Tree (DT), AdaBoost, Gradient Boosting and Voting classifiers were analyzed on basis of performance measures. After analysing the classifiers, the focus was to utilize the strengths of one model to complement the weakness of another. So, the multilevel voting model was proposed, which integrates various ML models based on their FP rates to retrieve a news voting classifier to retrieve better prediction analysis. The developed model helps to solve the trade-off issue between accuracy and efficiency.

In future, a web-based GUI will be created for the proposed fake news detection system to classify the news as fake or real on real-time social media platforms such as Facebook, Instagram, Twitter, WhatsApp, *etc.* Also, the annotated dataset in form of images (with textual content written on them) will be collected and maintained from Facebook and Reddit platforms. The annotated dataset can be used for detecting fake images in future as no such dataset is available at present. The proposed system has the potential to provide an impulse to various emerging applications such as controlling the spread of fake news during elections, terrorism, natural calamities, crimes for the betterment of the society.

Acknowledgements This Publication is an outcome of the R&D work undertaken in the project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

References

1. N.J. Conroy, V.L. Rubin, Y. Chen, Proceedings of the Association for Information Science and Technology **52**(1), 1 (2015)
2. Fake news on whatsapp. `http://bit.ly/2miuv9j`. Last accessed: 27-08-2019
3. K.R. Canini, B. Suh, P.L. Pirolli, in Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on (IEEE, 2011), pp. 1–8
4. H. Ahmed, I. Traore, S. Saad, in International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments (Springer, 2017), pp. 127–138
5. A. Gupta, H. Lamba, P. Kumaraguru, in eCrime Researchers Summit (eCRS), 2013 (IEEE, 2013), pp. 1–12
6. A.H. Wang, in Security and cryptography (SECRYPT), proceedings of the 2010 international conference on (IEEE, 2010), pp. 1–10
7. F. Benevenuto, G. Magno, T. Rodrigues, V. Almeida, in Collaboration, electronic messaging, anti-abuse and spam conference (CEAS), vol. 6 (2010), vol. 6, p. 12
8. I. Sen, A. Aggarwal, S. Mian, S. Singh, P. Kumaraguru, A. Datta, in Proceedings of the 10th ACM Conference on Web Science (ACM, 2018), pp. 205–209
9. K. Garimella, G. Tyson, arXiv preprint arXiv:1804.01473 (2018)
10. J.A. Caetano, J.F. de Oliveira, H.S. Lima, H.T. Marques-Neto, G. Magno, W. Meira Jr, V.A. Almeida, arXiv preprint arXiv:1804.00397 (2018)
11. T. Pontes, M. Vasconcelos, J. Almeida, P. Kumaraguru, V. Almeida, in Proceedings of the 2012 ACM conference on ubiquitous computing (ACM, 2012), pp. 898–905
12. P. Dewan, M. Gupta, K. Goyal, P. Kumaraguru, in Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop (ACM, 2013), p. 6
13. M. Weimer, I. Gurevych, M. Mühlhäuser, in Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions (Association for Computational Linguistics, 2007), pp. 125–128
14. W. Wei, X. Wan, arXiv preprint arXiv:1705.06031 (2017)
15. Y. Chen, N.J. Conroy, V.L. Rubin, in Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection (ACM, 2015), pp. 15–19
16. K. Shu, A. Sliva, S. Wang, J. Tang, H. Liu, ACM SIGKDD Explorations Newsletter **19**(1), 22 (2017)
17. V. Rubin, N. Conroy, Y. Chen, S. Cornwell, in Proceedings of the Second Workshop on Computational Approaches to Deception Detection (2016), pp. 7–17
18. P. Dewan, P. Kumaraguru, Social Network Analysis and Mining **7**(1), 15 (2017)
19. P. Jain, P. Kumaraguru, in Proceedings of the 3rd IKDD Conference on Data Science, 2016 (ACM, 2016), p. 6
20. B. Markines, C. Cattuto, F. Menczer, in Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web (ACM, 2009), pp. 41–48

21. F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, M. Gonçalves, in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (ACM, 2009), pp. 620–627
22. V. Qazvinian, E. Rosengren, D.R. Radev, Q. Mei, in Proceedings of the Conference on Empirical Methods in Natural Language Processing (Association for Computational Linguistics, 2011), pp. 1589–1599
23. V.L. Rubin, N.J. Conroy, Y. Chen, in Hawaii International Conference on System Sciences (2015)
24. A. Magdy, N. Wanas, in Proceedings of the 2nd international workshop on Search and mining user-generated contents (ACM, 2010), pp. 103–110
25. S. Volkova, K. Shaffer, J.Y. Jang, N. Hodas, in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2 (2017), vol. 2, pp. 647–653
26. S. Chhabra, A. Aggarwal, F. Benevenuto, P. Kumaraguru, in Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (ACM, 2011), pp. 92–101
27. A. Aggarwal, A. Rajadesingan, P. Kumaraguru, in eCrime Researchers Summit (eCrime), 2012 (IEEE, 2012), pp. 1–12
28. G. D'Angelo, F. Palmieri, S. Rampone, Information Sciences **486**, 31 (2019)
29. Y. LeCun, Y. Bengio, G. Hinton, nature **521**(7553), 436 (2015)
30. N. Ruchansky, S. Seo, Y. Liu, in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (ACM, 2017), pp. 797–806
31. F. Ahmed, M. Abulaish, in Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on (IEEE, 2012), pp. 602–608
32. S.M. SIRAJUDEEN, N.F.A. AZMI, A.I. ABUBAKAR, Journal of Theoretical & Applied Information Technology **95**(17) (2017)
33. H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, B.Y. Zhao, in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (ACM, 2010), pp. 35–47
34. P. Dewan, P. Kumaraguru, in Privacy, Security and Trust (PST), 2015 13th Annual Conference on (IEEE, 2015), pp. 85–92
35. A. Aggarwal, S. Kumar, K. Bhargava, P. Kumaraguru, (2018)
36. A. Gupta, P. Kumaraguru, Twitter explodes with activity in mumbai blasts! a lifeline or an unmonitored daemon in the lurking? Tech. rep. (2012)
37. A. Gupta, H. Lamba, P. Kumaraguru, A. Joshi, in Proceedings of the 22nd international conference on World Wide Web (ACM, 2013), pp. 729–736
38. A. Gupta, P. Kumaraguru, in Proceedings of the 1st workshop on privacy and security in online social media (ACM, 2012), p. 2
39. News trends database. `https://bit.ly/2zVRLxK`. Last accessed: 18-10-2017
40. Kaggle database. `https://bit.ly/2Ex5VsX`. Last accessed: 24-10-2017
41. Kaggle database. `https://bit.ly/2BmqBQE`. Last accessed: 22-10-2017
42. T. Pontes, G. Magno, M. Vasconcelos, A. Gupta, J. Almeida, P. Kumaraguru, V. Almeida, in Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on (IEEE, 2012), pp. 571–578
43. O. Batchelor, Reference Services Review **45**(2), 143 (2017)
44. A. Alahmadi, A. Joorabchi, A.E. Mahdi, in 2013 7th IEEE GCC Conference and Exhibition (GCC) (IEEE, 2013), pp. 108–113
45. S.Z. Mishu, S. Rafiuddin, in Computer and Information Technology (ICCIT), 2016 19th International Conference on (IEEE, 2016), pp. 409–413