

A NOVEL STACKING APPROACH FOR DETECTION OF FAKE NEWS

A PROJECT REPORT

Submitted in partial fulfillment of requirements to

ACHARYA NAGARJUNA UNIVERSITY

For the award of the Degree

Bachelor of Technology

in

INFORMATION TECHNOLOGY

By

M.TEJA VARDHAN (Y18IT103)

Y.SAI TEJA (Y18IT090)

J. ASHOK (L19IT125)



APRIL - 2022

R.V.R & J.C COLLEGE OF ENGINEERING(AUTONOMOUS)

(Affiliated to Acharya Nagarjuna University,GUNTUR)

(Approved by AICTE & Accredited by NBA)

Chandramoulipuram::Chowdavaram,GUNTUR-522019

**DEPARTMENT OF INFORMATION TECHNOLOGY
R.V.R & J.C COLLEGE OF ENGINEERING(AUTONOMOUS)**



BONAFIDE CERTIFICATE

This is to certify that this project work titled “**A NOVEL STACKING APPROACH FOR DETECTION OF FAKE NEWS**” is the bonafide work of **M.TEJA VARDHAN(Y18IT103), Y.SAI TEJA (Y18IT090), J.ASHOK (L19IT125)** who have carried out the work under my supervision, and submitted in partial fulfillment for the award of degree,**B.Tech** during the academic year 2021-22.

Mrs.B.MANASA

Guide

Assistant Professor

Information Technology

Dr.A.SRI KRISHNA

Head Of Department

Professor

Information Technology

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without proper suggestions, guidance and environment. The combination of these three factors act like a backbone to our Project “**A Novel Stacking Approach for Detection of Fake News**”.

We are very much thankful to our principal, **Dr.K.Ravindra**, Principal of R.V.R & J.C College of Engineering for providing the supportive environment.

We express our sincere thanks to **Dr.A.Sri Krishna**, Head of the Department of Information Technology for her encouragement and support to carry out this project successfully.

We are very glad to express our special thanks to **Mrs.B.Manasa**, Guide for the project, who has inspired us to select this topic, and also for her encouragement and valuable advice for doing this project.

We are very glad to express our special thanks to **Dr.M.Pompapathi**, Lecturer Incharge for the project who extended his encouragement and support to carry out this project successfully.

Finally, we submit our thanks to Lab Staff in the Department of Information Technology and to all our friends for their cooperation during the preparation.

M.Teja Vardhan (Y18IT103)

Y.Sai Teja (Y18IT090)

J.Ashok (L19IT125)

Table of Contents

Bonafide Certificate	ii
Acknowledgement	iii
Table of Contents	iv
Abstract	v
List of Figures	vi
List of Tables	vii
List of Abbreviations	vii
1. INTRODUCTION	
1.1 Background	1
1.2 Problem Definition	2
1.3 Significance of the work	2
1.4 Need for present study	2
2. LITERATURE REVIEW	
2.1 Introduction	3
2.2 Methodology	3
2.3 Results	7
3. PROPOSED WORK	
3.1 Architecture	13
3.2 Methodology	14
3.3 Machine Learning Classifiers	19
3.4 Deep Learning Models	24
3.5 Hold Out Cross Validation Method	28
3.6 Model Evaluation Criteria	28
4. RESULTS	29
5. CONCLUSION AND FUTURE WORK	34
REFERENCES	36

Abstract

With the increasing popularity of social media, people have changed the way they access news. News online has become the major source of information for people. However, much information appearing on the Internet is dubious and even intended to mislead. Some fake news are so similar to the real ones that it is difficult for humans to identify them. Therefore, automated fake news detection tools like machine learning and deep learning models have become an essential requirement. Here, the performance of five machine learning models and three deep learning models are evaluated on two fake and real news datasets of different size with hold out cross validation. The techniques like term frequency, term frequency-inverse document frequency and embedding are used to obtain text representation for machine learning and deep learning models respectively. To evaluate models' performance, accuracy, precision, recall and F1-score are used as the evaluation metrics and a corrected version of McNemar's test to determine if models' performance is significantly different. Then, the proposed novel stacking model which achieved testing accuracy of 99.94% on the ISOT dataset. Furthermore, the performance of the proposed method is high as compared to baseline methods.

List of Figures

1. Decision Tree	4
2. Random Forest	5
3. SVM	6
4. Architecture Of Proposed Work	14
5. Fake News Data set	15
6. Real News Data set	15
7. Explanation of Glove	18
8. Explanation of One Hot Vector	19
9. Comparison of linear and logistic	20
10. KNN	22
11. SVM-Finding the optimal hyper line	23
12. Matrix of pixel value	25
13. LSTM	26
14. GRU	27
15. Comparison between LSTM & GRU	28

List of Tables

1.Results	12
2. Stemmed Text results	30
3. Lemmatized Text results	30

List of Abbreviations

SVM - Support Vector Machine

KNN - K-Nearest Neighbor

DT - Decision Tree

RF - Random Forest

CNN - Convolutional Neural Network

GRU - Gated Recurrent Network

LSTM - Long Short- Term Memory

TF - Term Frequency

TF-IDF - Term Frequency-Inverse Document Frequency

CHAPTER 1

INTRODUCTION

1.1 Background

With the rapid development of the Internet, social media has become a perfect platform for spreading fake news, distorted information, fake reviews, rumors, satires. Many people think the 2016 U.S. presidential election campaign has been influenced by fake news. Subsequent to this election, the term has entered the mainstream vernacular .

Nowadays fake news has become a major concern for both industry and academia, one of the solutions for this problem is human fact-checking. However, the real-time nature of fake news on social media makes identifying fake news even more difficult . The expert fact checking may have very limited help because of its low efficiency. Moreover, fact-checking by humans is Expensive. Thus, There is a need to use Machine Learning (ML) and Deep Learning Models (DL) to automate this process. Various classification can be used for fake news detection.

The evaluation of different classification algorithms such as logistic regression (LR) supports vector machine (SVM), k-nearest neighbor (k-NN),decision tree (DT),random forest(RF), convolutional neural network (CNN),gated recurrent network (GRU) , long short term memory (LSTM)for the detection of Fake news. Then a stacking method to improve the individual model performance.Techniques like term frequency (TF), term frequency-inverse document frequency (TF-IDF) and embedding to tokenize the title and text feature of these two datasets.

Grid Search technique has been used for tuning the hyper parameters and model selection. Various performance evaluation metrics have been used such as accuracy, recall, f1-score, precision and training time. The experimental results of the proposed stacking method have been compared with the state of the art results in the published literature. Furthermore, all experimental results have been tabulated in various tables and graphically shown in various figures for better understanding.

The paper have the following contributions:

- Firstly, five machine learning models and three deep learning models have been trained to compare the performance difference between individual models.
- Secondly, two datasets of different sizes are used to test models' robustness on datasets of different sizes.
- Thirdly, employ a corrected version of McNemar's statistical test to decide if there really are significant differences between two model's performance and choose the best individual model

for fake news detection.

- Lastly, our proposed stacking model outperformed the state of the art methods.

1.2 Problem Definition

Fake news detection is a global problem, different fake news in different countries is written in different languages. Researchers try to find a solution for multiple language fake news detection by constructing a new dataset or training models on different language datasets. Faustini et al. trained Naïve Bayes, K-Nearest Neighbors, SVM and Random Forest from five datasets in three languages. They compared the results obtained through a custom set of features, Document-class Distance, bag-of-words and Word2Vec in accuracy and F1-Score measures. Eventually, they concluded that SVM and RandomForest outperformed other algorithms and bag-of-words achieved the best results in general. Wang et al. presented an English fake news dataset. They also designed a hybrid CNN model to integrate metadata with text and proved that this hybrid approach can improve a text-only model

1.3 Significance of the work

Different classification algorithms such as logistic regression (LR), support vector machine (SVM), k-nearest neighbor (k-NN), decision tree (DT), random forest (RF), convolutional neural network (CNN), gated recurrent network (GRU), long short-term memory (LSTM) are evaluated for the detection of Fake news. Then used a stacking method to improve the individual model performance. ISOT dataset. The techniques like term frequency (TF), term frequency-inverse document frequency (TF-IDF) and embedding are used to tokenize the title and text feature of these two datasets. Grid Search technique has been used for tuning the hyper parameters and model selection. Various performance evaluation metrics have been used such as accuracy, recall, f1-score, precision and training time. The experimental results of the proposed stacking method have been compared with the state of the art results in the published literature. Furthermore, all experimental results have been tabulated in various tables and graphically shown in various figures for better understanding.

1.4 Need for Present study

The main aim of this project is to detect fake news. Numerous machine learning and deep Learning techniques have been recommended by various scholars. Hence, some of the base lines of fake news detection techniques are presented. The major objectives of the literature review is to identify the problems in the baseline methods and provide a reliable solution.

CHAPTER 2

LITERATURE SURVEY

1. Fake News Detection using Machine Learning Approach.(Authors:Z Khanam¹, B N Alwasel, H Sirafi¹ and M Rashid)

2.1 Introduction

Fake News contains misleading information that could be checked. This maintains a lie about a certain statistic in a country or exaggerated cost of certain services for a country, which may cause unrest for some countries like in the Arab spring. There are organizations, like the House of Commons and the Crosscheck project, trying to deal with issues as confirming authors are accountable. However, their scope is so limited because they depend on human manual detection. In a globe with millions of articles either removed or being published every minute, this cannot be accountable or feasible manually. A solution could be, by the development of a system to provide a credible automated index scoring, or rating for credibility of different publishers, and news context.

This paper proposes a methodology to create a model that will detect if an article is authentic or fake based on its words, phrases, sources and titles, by applying supervised machine learning algorithms on an annotated (labeled) dataset, that are manually classified and guaranteed. Then, feature selection methods are applied to experiment and choose the best fit features to obtain the highest precision, according to confusion matrix results. Therefore propose to create the model using different classification algorithms. The product model will test the unseen data, the results will be plotted, and accordingly, the product will be a model that detects and classifies fake articles and can be used and integrated with any system for future use.

2.2 Methodologies

2.2.1 Naive Bayes

This algorithm works on Bayes theorem under the assumption that it is free from predictors and is used in multiple machine learning problems [18]. Simply put, Naive Bayes assumes that one function in the category has nothing to do with another. For example, the fruit will be classified as an apple when it's of red color, swirls, and the diameter is close to 3 inches. Regardless of whether these functions depend on each other or on different functions, and even if these functions depend on each other or on other functions, Naive Bayes assumes that all these functions share a separate proof of the apples [14]

Naive Bayes Equation

$$P(c/x) = P(x/c) * P(c) / P(x)$$

$$P(c/X) = P(x_1/c) * P(x_2/c) \dots P(x_n/c) * P(c)$$

Where:

$P(c/X)$ is the posterior Probability.

$P(x/c)$ is the Likelihood.

$P(c)$ is the Class Prior Probability.

$P(x)$ is the Predictor Prior Probability.

2.2.2 Decision Tree

The decision tree is an important tool that works based on flow chart-like structure that is mainly used for classification problems. Each internal node of the decision tree specifies a condition or a “test” on an attribute (as shown in Fig.1) and the branching is done on the basis of the test conditions and result.

Finally, the leaf node bears a class label that is obtained after computing all attributes. The distance from the root to leaf represents the classification rule. The amazing thing is that it can work with category and dependent variables. They are good in identifying the most important variables and they also depict the relation between the variables quite aptly. They are significant in creating new variables and features which are useful for data exploration and predict the target variable quite efficiently.

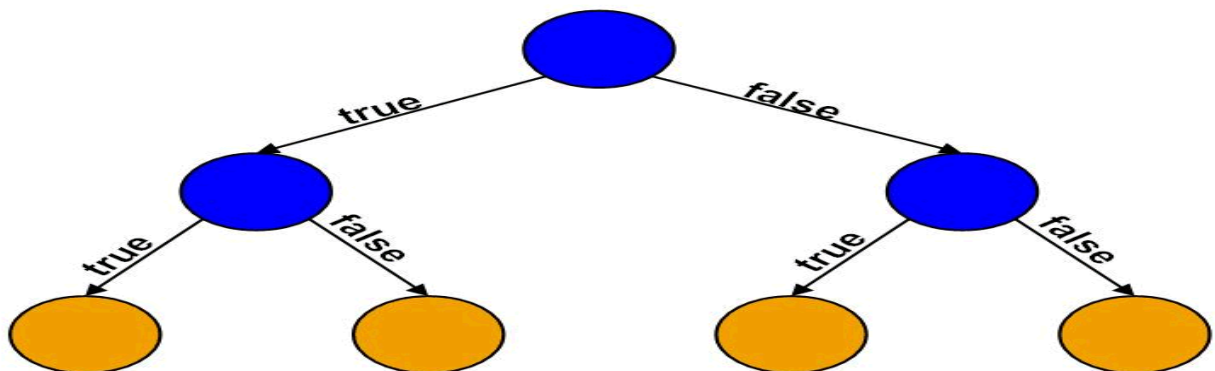


Fig 1. Decision Tree

2.2.3 Random Forest

Random Forests are built on the concept of building many decision tree algorithms, after which the decision trees get a separate result. The results, which are predicted by a large number of decision trees, are taken up by the random forest(as shown in Fig.2). To ensure a variation of the decision trees, the random forest randomly selects a subcategory of properties from each group [16][17].

The applicability of Random forest is best when used on uncorrelated decision trees. If applied on similar trees, the overall result will be more or less similar to a single decision tree. Uncorrelated decision trees can be obtained by bootstrapping and feature randomness.

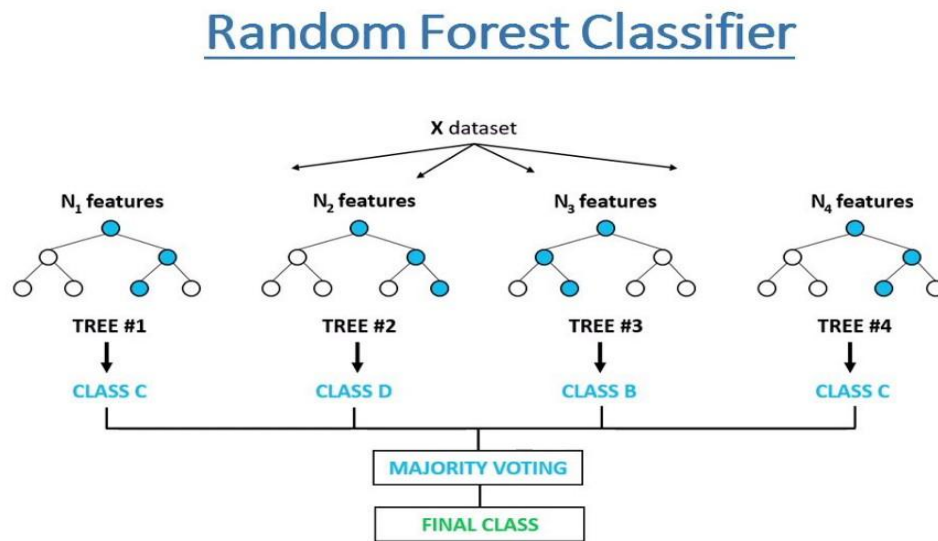


Fig 2. Random Forest

2.2.4 Support Vector Machine

The SVM algorithm is based on the layout of each data item in the form of a point in a range of dimensions (the number of available properties), and the value of a given property is the number of specified coordinates [13]. Given a set of n features, SVM algorithm uses n dimensional space to plot the data item with the coordinates representing the value of each feature. The hyper-plane obtained to separate the two classes(as shown in Fig.3) is used for classifying the data.

SVM is a discriminative classifier formally defined by a separating hyperplane. Its main task is to select a hyperplane with the maximum possible margin between the support vectors in the given dataset. Hyperplane is a decision plane that separates between a set of objects having different class memberships. Margin is the perpendicular distance between the hyperplane and the support vectors.

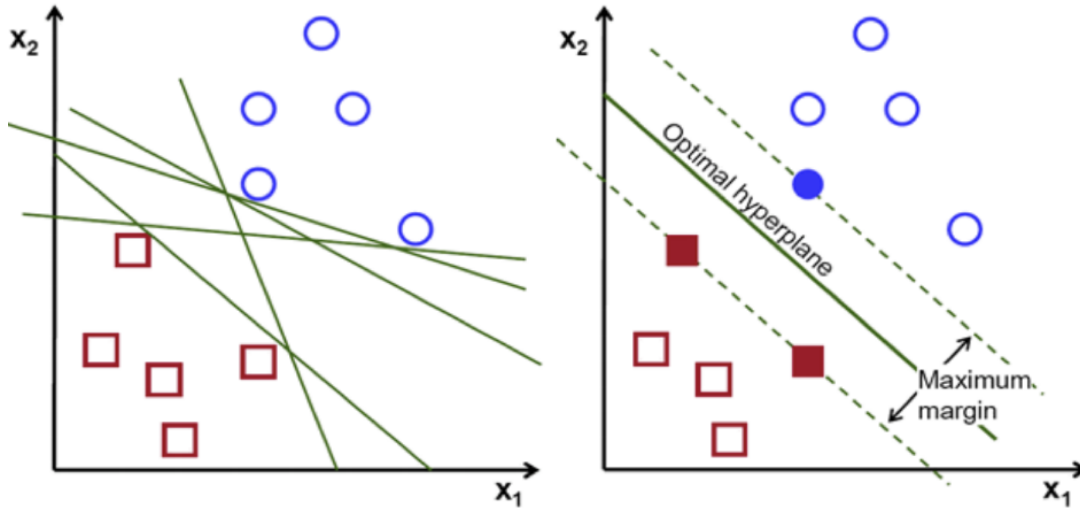


Fig 3. SVM

2.2.5 Logistic Regression

Logistic Regression estimates the probability that a data item belongs to a particular category. Here, the logistic function or sigmoid function is used to model the probabilities. The sigmoid function gives output between 0 and 1 for all the input values. The logistic function will always produce an S-shaped curve regardless of the input variable X . The below equation (3.1) can be reframed as

$$\log \left(\frac{p(X)}{(1 - p(X))} \right) = b_0 + b_1 * X \quad (3.1)$$

The quantity $\frac{p(X)}{(1 - p(X))}$ is called the odds ratio, and can take any value between 0 and ∞ .

Values of the odds ratio very close to 0 and ∞ indicate very low and very high probabilities of $p(X)$

respectively. The quantity $\log \left(\frac{p(X)}{(1 - p(X))} \right)$ is called the logit.

The Coefficient b_0 and b_1 are unknown and must be estimated based on the available training data. For this purpose, the maximum likelihood, a powerful statistical technique. Maximum likelihood is a very good approach for fitting non-linear models.

The mathematical function (3.2) for the likelihood can be given as:

$$l(b_0, b_1) = p(X) * (1 - p(X)) \quad (3.2)$$

2.3 Results

Naive Bayes gives an accuracy of 95.01%(mentioned in table.1)

▾ Naive Bayes

```
✓ 9s ▶ dct = dict()

from sklearn.naive_bayes import MultinomialNB

NB_classifier = MultinomialNB()
pipe = Pipeline([('vect', CountVectorizer()),
                  ('tfidf', TfidfTransformer()),
                  ('model', NB_classifier)])

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}".format(round(accuracy_score(y_test, prediction)*100,2)))

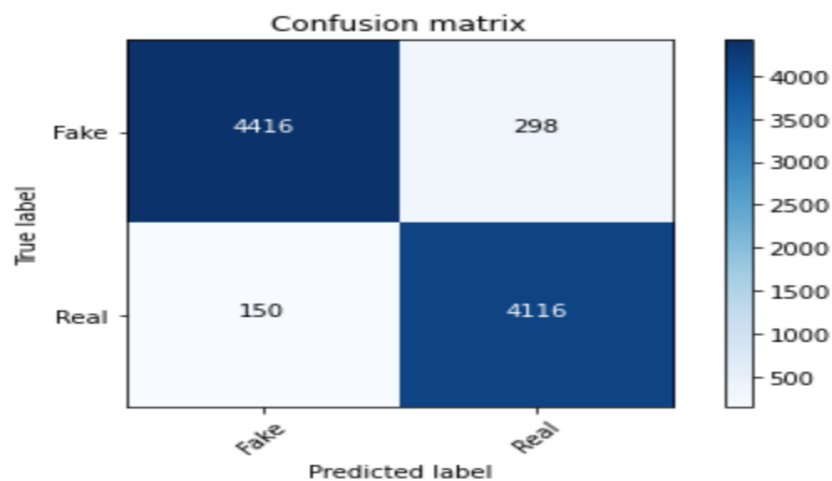
dct['Naive Bayes'] = round(accuracy_score(y_test, prediction)*100,2)
```

📄 accuracy: 95.01%

+ Code

```
[26] cm = metrics.confusion_matrix(y_test, prediction)
      plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

Confusion matrix, without normalization



▼ Logistic regression

```
✓ 16s ▶ # Vectorizing and applying TF-IDF
from sklearn.linear_model import LogisticRegression

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

# Fitting the model
model = pipe.fit(X_train, y_train)

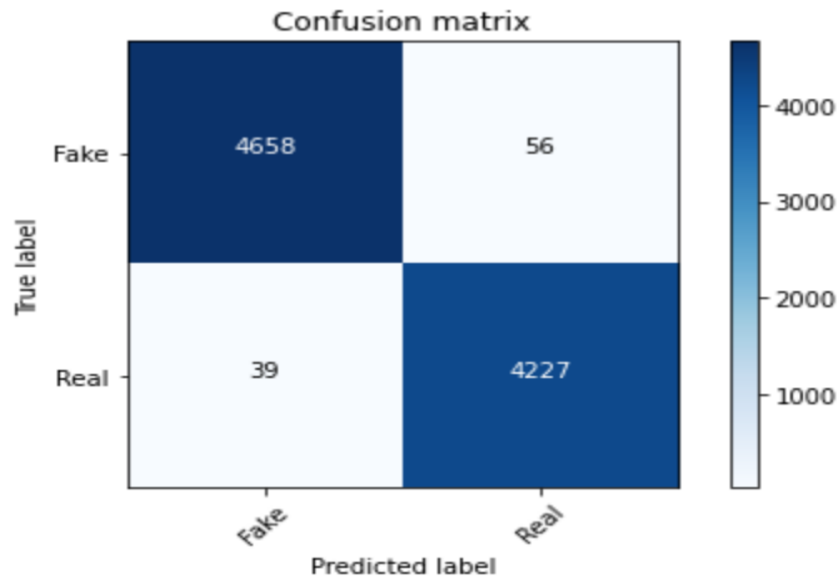
# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Logistic Regression'] = round(accuracy_score(y_test, prediction)*100,2)
```

☞ accuracy: 98.94%

Logistic Regression gives an accuracy of 98.94%(mentioned in table.1)

```
✓ 0s [28] cm = metrics.confusion_matrix(y_test, prediction)
      plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

Confusion matrix, without normalization



▼ Decision Tree

✓
28s

```
from sklearn.tree import DecisionTreeClassifier

# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                  ('tfidf', TfidfTransformer()),
                  ('model', DecisionTreeClassifier(criterion= 'entropy',
                                                    max_depth = 20,
                                                    splitter='best',
                                                    random_state=42))])

# Fitting the model
model = pipe.fit(X_train, y_train)

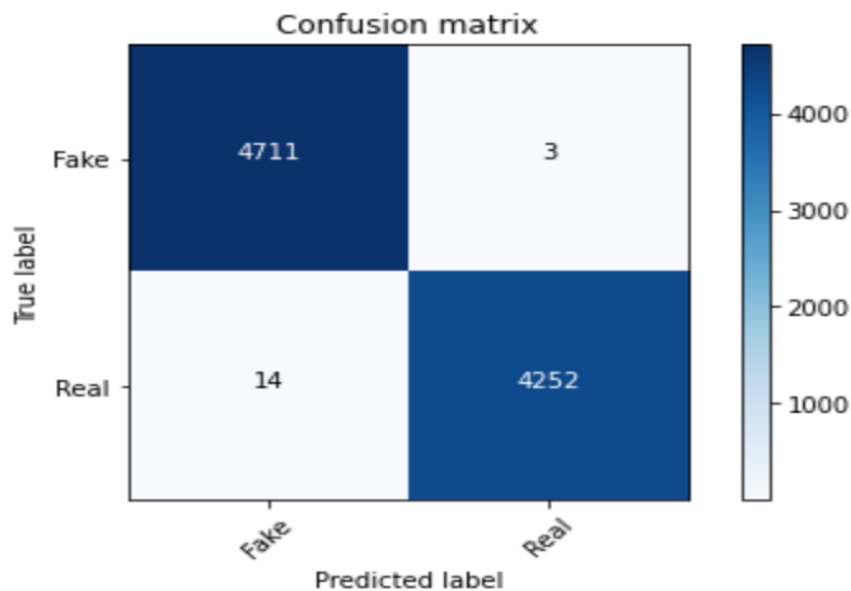
# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Decision Tree'] = round(accuracy_score(y_test, prediction)*100,2)
```

➞ accuracy: 99.81%

Decision Tree gives an accuracy of 99.81%(mentioned in table.1)

```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=[ 'Fake', 'Real' ])
```

➞ Confusion matrix, without normalization



▼ Random Forest

```
[31] from sklearn.ensemble import RandomForestClassifier

pipe = Pipeline([('vect', CountVectorizer()),
                  ('tfidf', TfidfTransformer()),
                  ('model', RandomForestClassifier(n_estimators=50, criterion="entropy"))])

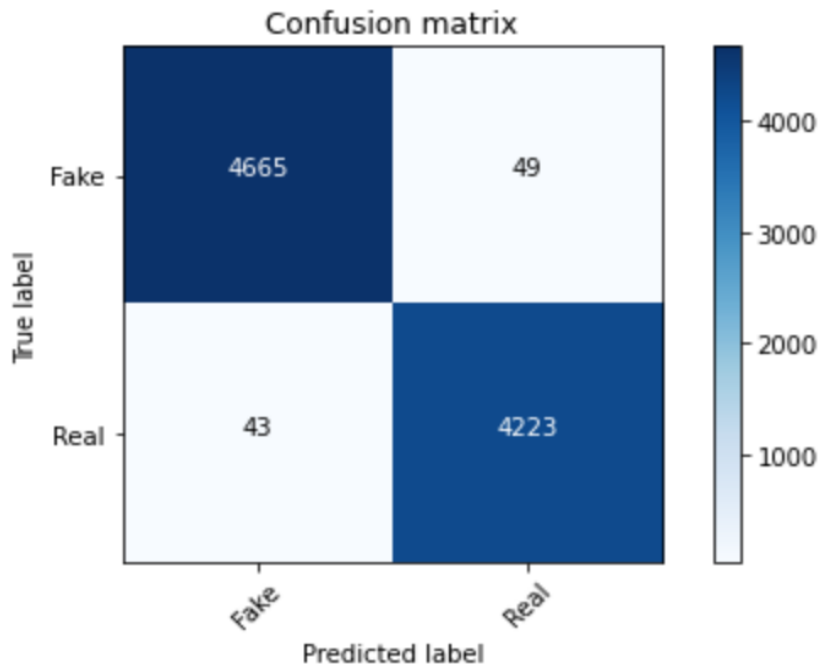
model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Random Forest'] = round(accuracy_score(y_test, prediction)*100,2)

accuracy: 98.98%
```

Random Forest gives an accuracy of 98.98%(mentioned in table.1)

```
[32] cm = metrics.confusion_matrix(y_test, prediction)
      plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

Confusion matrix, without normalization



▼ SVM

```
[33] from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

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

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['SVM'] = round(accuracy_score(y_test, prediction)*100,2)
```

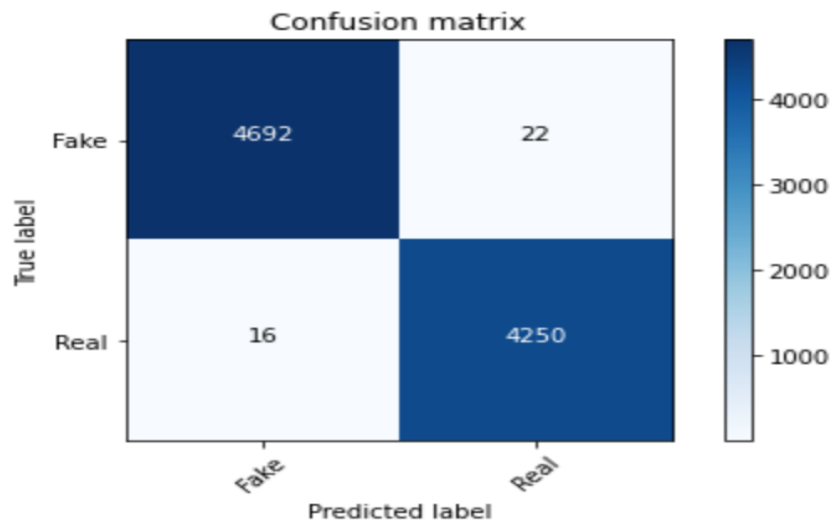
accuracy: 99.58%

SVM gives an accuracy of 99.58%(mentioned in table.1)

+ Code

```
[34] cm = metrics.confusion_matrix(y_test, prediction)
      plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

Confusion matrix, without normalization



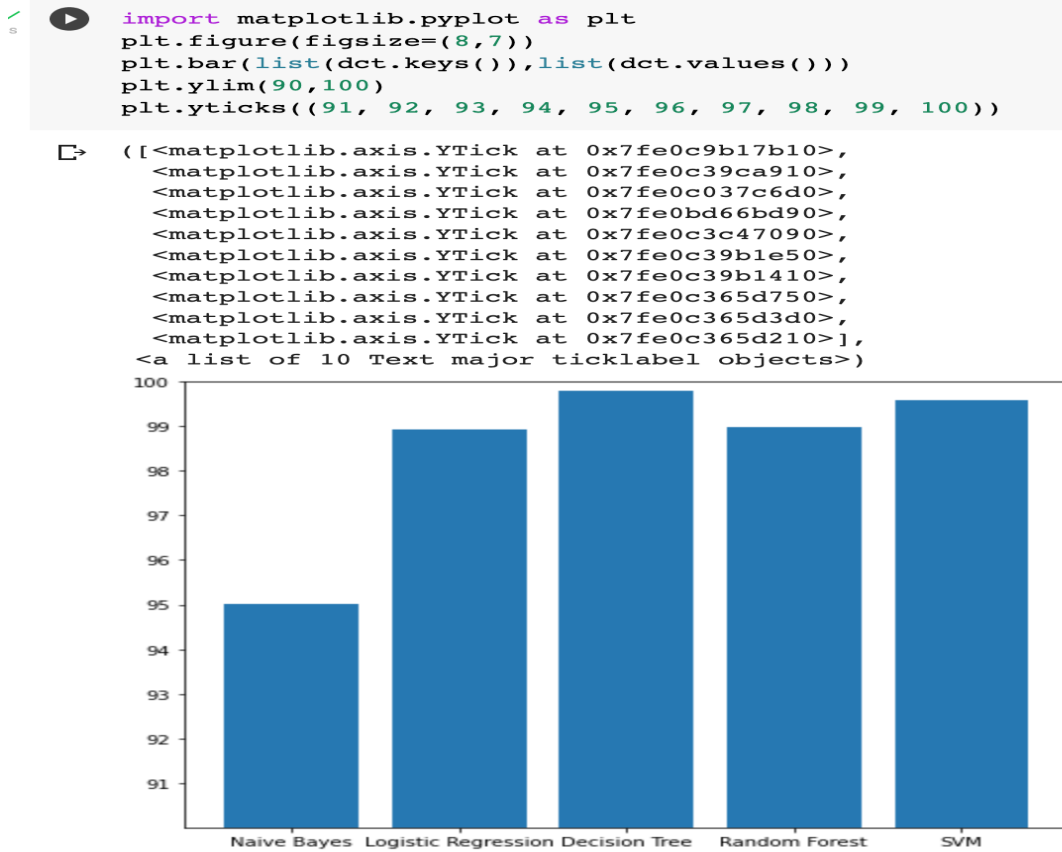


Table 1: Results

Method	Accuracy
Naive Bayes	95.01
Decision Tree	99.81
Random Forest	98.98
SVM	99.58
Logistic Regression	98.94

CHAPTER 3

PROPOSED WORK

3.1 Architecture

The advent of the World Wide Web and the rapid adoption of social media platforms (such as Facebook and Twitter) paved the way for information dissemination that has never been witnessed in human history before. With the current usage of social media platforms, consumers are creating and sharing more information than ever before, some of which are misleading with no relevance to reality. Automated classification of a text article as misinformation or disinformation is a challenging task. Even an expert in a particular domain has to explore multiple aspects before giving a verdict on the truthfulness of an article. In this work, a machine learning ensemble approach is proposed for automated classification of news articles. Our study explores different textual properties that can be used to distinguish fake contents from real. By using those properties, a combination of different machine learning and DL algorithms are trained and evaluate their performance on real world datasets.

The architecture of the proposed system(Fig.4 below) gives the basic idea about the tasks that are to be performed. The architecture consists of the different stages that are to be considered. Firstly, need to have a dataset. The data is then pre-processed and different Machine Learning classifiers like Logistic Regression,Support vector Machine,Random Forest and so on are applied on that data. After the machine learning classifiers, Deep Learning Classifiers are performed on the data. Results can be obtained by performing the testing. Once the results are obtained, they can be analyzed accordingly.

The modules to be implemented are

1. Importing Dataset
2. Data Pre-Processing
3. Machine Learning Classifiers
4. Deep Learning Classifiers
- 5.Validation and Model Evaluation

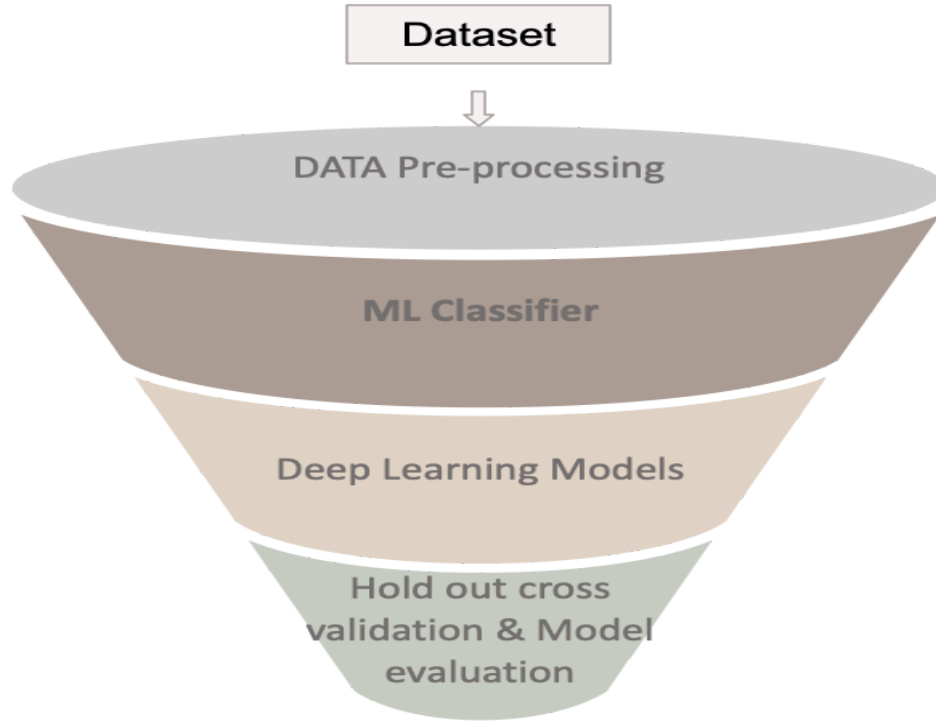


Figure 4. Architecture Of Proposed Work.

3.2 Methodology

3.2.1 DataSet

There are two datasets with Real News data and Fake News data. Each dataset has around 22000 articles. By merging both, around 45000 articles of real and fake are formed. The aim is to build a model to correctly predict if a news is real or fake. Each dataset contains Four attributes such as Title, Text, Subject, and Date.

The ISOT dataset was entirely collected from real-world sources. The real news was collected by crawling articles from Reuters.com and fake news were collected from unreliable websites that were flagged by Politifact and Wikipedia. The articles were mostly released from 2016 to 2017. This dataset(Fig.5 & Fig.6 below) includes 44898 data in total, 21417 are real news (labeled as 1) and 23481 are fake news(labeled as 0). The features include title, text (news body), subject, date and label.

The news has different categories like ‘politicsNews’, ‘worldnews’, ‘News’, ‘politics’, ‘Government News’, ‘left-news’, ‘US_News’, ‘Middle-east’. Here features like title and text in ISOT dataset to train our models.

	A	B	C	D	E
1	title	text	subject	date	
2	Donald Tru	Donald Tru	News	31-Dec-17	
3	Drunk Brag	House Intell	News	31-Dec-17	
4	Sheriff Davi	On Friday, it	News	30-Dec-17	
5	Trump Is Sc	On Christmas	News	29-Dec-17	
6	Pope Franci	Pope Franci	News	25-Dec-17	
7	Racist Alab	The number	News	25-Dec-17	
8	Fresh Off T	Donald Tru	News	23-Dec-17	
9	Trump Said	In the wake	News	23-Dec-17	
10	Former CIA	Many peopl	News	22-Dec-17	
11	WATCH: Br	Just when y	News	21-Dec-17	
12	Papa John's	A centerple	News	21-Dec-17	
13	WATCH: Pa	Republicans	News	21-Dec-17	
14	Bad News f	Republicans	News	21-Dec-17	
15	WATCH: Li	The media f	News	20-Dec-17	
16	Helena To	Abigail Dis	News	20-Dec-17	
17	Tone Deaf	Donald Tru	News	20-Dec-17	
18	The Intern	A new anim	News	19-Dec-17	
19	Mueller Sp	Trump supp	News	17-Dec-17	
20	SNL Hilar	Right now, t	News	17-Dec-17	
21	Republican	Senate Maj	News	16-Dec-17	
22	In A Heart	It almost s	News	16-Dec-17	
23	KY GOP Sta	In this #M	News	13-Dec-17	
24	Meghan Mc	As a Democ	News	12-Dec-17	
25	CNN CALLS	Alabama is	News	12-Dec-17	
26	White Hous	A backdash	News	12-Dec-17	
27	Despicable	Donald Tru	News	12-Dec-17	
28	Accused Ch	Ronald Roa	News	11-Dec-17	
29	WATCH: Fo	Judge Jean	News	10-Dec-17	
30	Liberal Gro	Donald Tru	News	09-Dec-17	

Figure 5. Fake News Data set

1	title	text	subject	date
2	As U.S. budg	WASHINGTON	politicsNew	31-Dec-17
3	U.S. military	WASHINGTON	politicsNew	29-Dec-17
4	Senior U.S. i	WASHINGTON	politicsNew	31-Dec-17
5	FBI Russia p	WASHINGTON	politicsNew	30-Dec-17
6	Trump want	SEATTLE/W	politicsNew	29-Dec-17
7	White Hous	WEST PALM	politicsNew	29-Dec-17
8	Trump says	WEST PALM	politicsNew	29-Dec-17
9	Factbox: Tr	The followin	politicsNew	29-Dec-17
10	Trump on T	The followin	politicsNew	29-Dec-17
11	Alabama of	WASHINGTON	politicsNew	28-Dec-17
12	Jones certifi	(Reuters) - A	politicsNew	28-Dec-17
13	New York g	NEW YORK/	politicsNew	28-Dec-17
14	Factbox: Tr	The followin	politicsNew	28-Dec-17
15	Trump on T	The followin	politicsNew	28-Dec-17
16	Man says he	(In Dec. 25	politicsNew	25-Dec-17
17	Virginia offi	(Reuters) - A	politicsNew	27-Dec-17
18	U.S. lawmal	WASHINGTON	politicsNew	27-Dec-17

Figure 6. Real News Data set

3.2.2 Data Preprocessing

Before the data were fed into machine learning and deep learning models, the text data needed to be preprocessed using methods like stop word removal, tokenization, sentence segmentation, and punctuation removal. These operations can significantly help us select the most relevant terms and increase model performance.

Both the datasets come from real word news articles, so there are a lot of meaningless urls which carry no information. So the data is cleaned first first by removing these urls. Stop word removal is the next preprocessing step. Stop words frequently used in English sentences to complete the sentence structure but they are insignificant in expressing an individual's thoughts. So in all the experiments, they removed one in case they created too much noise. After the text data was cleaned, tokenized them by using TF-IDF and embedding techniques.

The data can be classified only after obtaining the clean text.

- **Removal of URLs, Mentions and Punctuations:**
The social media posts usually contain URLs and mentions. These URLs are the web addresses which act as a reference to a web resource. An example URL is "https://www.google.com". Mentions start with the character"@". Punctuations are the marks which are used to separate the sentences and their elements to clarify meaning. Some of the punctuation marks are comma, full stop, apostrophe, etc. The clean text should be free from these URLs, mentions and the punctuations.
- **Removal of stop words:**
Stop words are the most commonly used words in a language. Stop words should be removed before processing the natural language data. Some of the example stop words in English are a, an, the, i, me, etc. The list of stop words can be obtained from the natural language toolkit library for the python language. Stop words in the text can be removed by using that list.
- **Stemming:**
Stemming is the process of converting the words into their root form. This step can be performed by using either Stemming or Lemmatization. Lemmatization returns root forms of actual language. For example, consider the word studies. Stemming returns studi, whereas lemmatization returns study. Consider another word better. Stemming returns better, whereas lemmatization returns good. So, for better results first lemmatization is applied and then stemming is performed.

Term frequency (TF)

TF is a common tokenization technique that calculates the similarity between documents by using the counts of words in the documents. By utilizing TF technique, each document will be represented by a vector that contains the word counts. Then each vector will be normalized and the sum of its elements will be one which makes the word counts convert into probabilities.

Term frequency-inverse document frequency (TF-IDF)

In the machine learning experiments, TF-IDF is used to transform the data into vectors. TF-IDF is a weighting metric commonly used in text classification problems. It is used to assign a score which shows the importance of the term to every term in the document. In this method, a term's significance increases with the frequency of the term in the dataset.

Embedding

In deep learning experiments, a trained word embedding technique is used to obtain text representation. In word embedding space, the geometrical distance between word vectors represents the semantic relationship. Word embedding can project the real human language grammar into a vector space. In an ideal embedding space, words sharing the same semantic meaning will be embedded into similar vectors. The geometrical distance between two word vectors is highly associated with their linguistic meaning. To reduce trainable parameters and increase time efficiency, Then used a file of Glove word embeddings .

The embedding techniques utilized include:

Word2vect:

A word embedding is an approach used to provide dense vector representation of words that capture some context words about their own. These are improved versions of simple bag-of-words models like word counts and frequency counters, mostly representing sparse vectors.

Word embeddings use an algorithm to train fixed-length dense vectors and continuous-valued vectors based on a large text corpus. Each word represents a point in vector space, and these points are learned and moved around the target word by preserving semantic relationships. The vector space representation of words provides a projection where words with similar meanings are clustered within the space.

The use of embeddings over the other text representation techniques like one-hot encodes, TF-IDF Bag-of-Words is one of the key methods which has led to many outstanding performances on deep neural networks with problems like neural machine translations. Moreover, some word embedding algorithms like GloVe and word2vec are likely to produce a state of performance achieved by neural networks.

GloVe(Fig.7 below) stands for Global Vectors for word representation. It is an unsupervised learning algorithm developed by researchers at Stanford University aiming to generate word embeddings by aggregating global word co-occurrence matrices from a given corpus.

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

The co-occurrence matrix for the sentence “the cat sat on the mat” with a window size of 1

Figure 7. Explanation of GloVe

The basic idea behind the GloVe word embedding is to derive the relationship between the words from statistics. Unlike the occurrence matrix, the co-occurrence matrix tells you how often a particular word pair occurs together. Each value in the co-occurrence matrix represents a pair of words occurring together. This is the idea behind the GloVe pre-trained word embeddings(3.3), and it is expressed as;

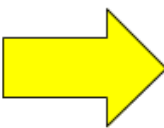
$$F(w_i, w_j, \tilde{w}_k) = P_{ik} - (1) P_{jk} \quad (3.3)$$

One Hot Encoding:

As a machine can only understand numbers and cannot understand the text in the first place, this essentially becomes the case with Deep Learning & ML algorithms. One hot encoding(Fig.8 below) can be defined as the essential process of converting the categorical data variables to be provided to machine and deep learning algorithms which in turn improve predictions as well as classification accuracy of a model. One Hot Encoding is a common way of preprocessing categorical features for machine learning models. This type of encoding creates a new binary feature for each possible category and assigns a value of 1 to the feature of each sample that corresponds to its original category.

One hot encoding is a highly essential part of the feature engineering process in training for learning techniques. For example, variables like colors and the labels were “red,” “green,” and “blue,” Encode each of these labels as a three-element binary vector as Red: [1, 0, 0], Green: [0, 1, 0], Blue: [0, 0, 1].

Color			
Red			
Red			
Yellow			
Green			
Yellow			



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Figure 8. Explanation of One Hot Vector

The Categorical data while processing, must be converted to a numerical form. One-hot encoding is generally applied to the integer representation of the data. Here the integer encoded variable is removed and a new binary variable is added for each unique integer value. During the process, it takes a column that has categorical data, which has been label encoded and then splits the following column into multiple columns. The numbers are replaced by 1s and 0s randomly, depending on which column has what value. While the method is helpful for some ordinal situations, some input data does not have any ranking for category values, and this can lead to issues with predictions and poor performance.

The One Hot Encoding technique creates a number of additional features based on the number of unique values in the categorical feature. Every unique value in the category is added as a feature. Hence the One Hot Encoding is known as the process of creating dummy variables. This technique helps create better classifiers and is very effective when combined in Deep Learning models.

3.3 Machine Learning Classifier

To detect the fake news , classifying approaches are employed to estimate the Fake News. The proposed framework is developed using Logistic Regression [1], [50], Support Vector Machine [61], Random Forest [8], Decision Tree [46], and KNN [21], [25].

3.3.1 Logistic regression

Logistic Regression estimates the probability that a data item belongs to a particular category. Here, the logistic function or sigmoid function(as shown in Fig.9) to model the probabilities. The sigmoid function gives output between 0 and 1 for all the input values.

$$p(X) = \frac{e^{(b_0 + b_1 * X)}}{1 + e^{(b_0 + b_1 * X)}} \quad (3.4)$$

The logistic function will always produce an S-shaped curve regardless of the input variable X . The above equation(3.4) can be reframed as

$$\log\left(\frac{p(X)}{(1 - p(X))}\right) = b_0 + b_1 * X \quad (3.5)$$

The quantity $\frac{p(X)}{(1 - p(X))}$ is called the odds ratio, and can take any value between 0 and ∞ . Values of the odds ratio very close to 0 and ∞ indicate very low and very high probabilities of $p(X)$

respectively. The quantity $\log\left(\frac{p(X)}{(1 - p(X))}\right)$ is called the logit.

The Coefficient b_0 and b_1 are unknown and must be estimated based on the available training data. For this purpose, the maximum likelihood, a powerful statistical technique. Maximum likelihood is a very good approach for fitting non-linear models.

The mathematical function(3.6) for the likelihood can be given as:

$$l(b_0, b_1) = p(X) * (1 - p(X)) \quad (3.6)$$

The estimated b_0 and b_1 are chosen to maximize the likelihood function. Once the coefficients have been estimated, calculate the probability $p(X)$, which yields a value 1 for depressed individuals and 0 for non-depressed individuals.

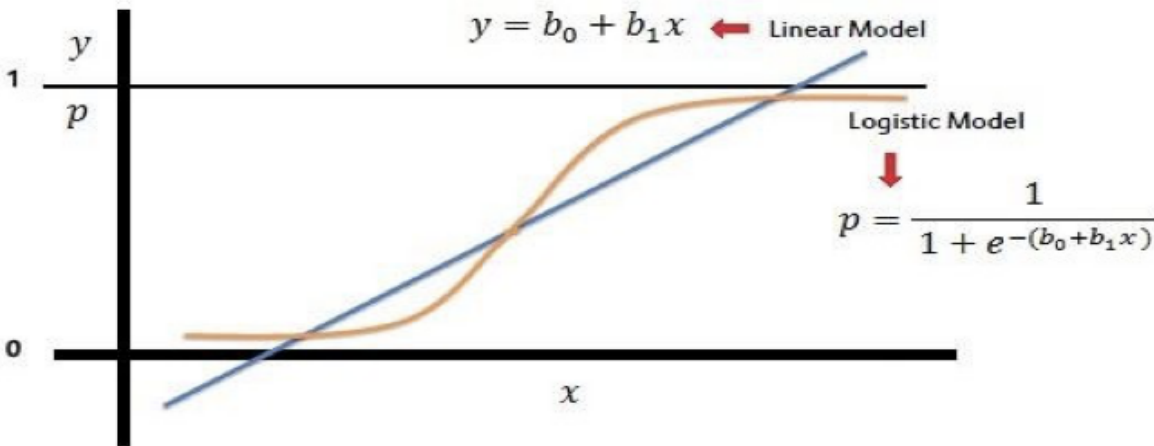


Figure 9. Comparison of linear model and logistic model

3.3.2 Decision Tree

The basic idea of DT is that it develops a model to predict the value of a dependent factor by learning various decision rules inferred from the whole data. Decision trees are upside down which means the root is at the top and then this root is split into various several nodes. Decision trees are nothing but a bunch of if-else statements in layman terms. It checks if the condition is true and if it is then it goes to the next node attached to that decision.

Entropy:

Entropy is nothing but the uncertainty in our dataset or measure of disorder.

The formula for Entropy(3.7) is shown below:

$$E(S) = -p_{(+)} \log p_{(+)} - p_{(-)} \log p_{(-)} \quad (3.7)$$

Here, p_{+} is the probability of positive class

p_{-} is the probability of negative class

S is the subset of the training example

$$Gain(A, D) = Entropy(D) - \sum_{i=1}^n \frac{|D_i|}{|D|} Entropy(D_i) \quad (6)$$

$$GainRatio(A, D) = \frac{Gain(D, A)}{IV(A)} \quad (7)$$

where intrinsic value of attribute A can be calculated as:

$$IV(A) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|} \quad (8)$$

22631

3.3.3 K-Nearest Neighbor

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity (as shown in Fig.10). This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.

KNN Algorithm

- 1: for all unlabeled data u do
- 2: for all labeled data v do
- 3: compute the distance between u and v
- 4: find k smallest distances and locate the corresponding labeled instances v1, . . . vk
- 5: assign unlabeled data u to the label appearing most frequently in the located labeled
- 6: end for
- 7: end for
- 8: End

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

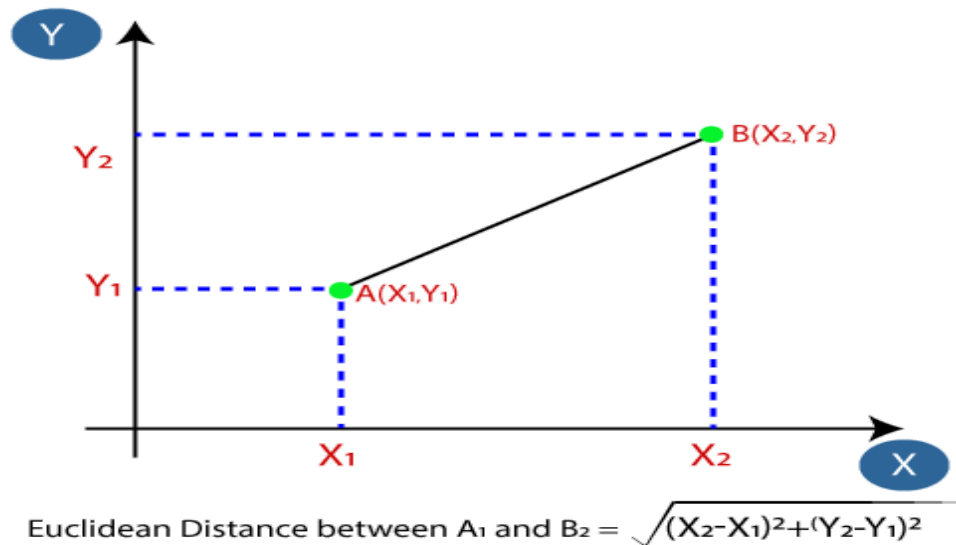


Figure 10. KNN

3.3.4 Support Vector Machine

The main objective of SVM is to select a hyperplane with the maximum possible margin between the support vectors in the given dataset. Support vectors are the data points which are closest to the hyperplane. Hyperplane is a decision plane that separates between a set of objects having different class memberships (as shown in Fig. 11). Margin is the perpendicular distance between the hyperplane and the support vectors. Generate the hyperplanes which segregate the classes in the best way.

Then, identify and select the right hyperplane that classifies the data points correctly and has the good margin.

The SVM algorithm is implemented by using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called kernel trick, which takes a low- dimensional input space and transforms it into a higher dimensional space. It is most useful in the non-linear separation problem.

A linear kernel(3.8) can be used as a normal dot product of any two given

$$K(x, x_i) = \text{sum}(x * x_i) \quad (3.8)$$

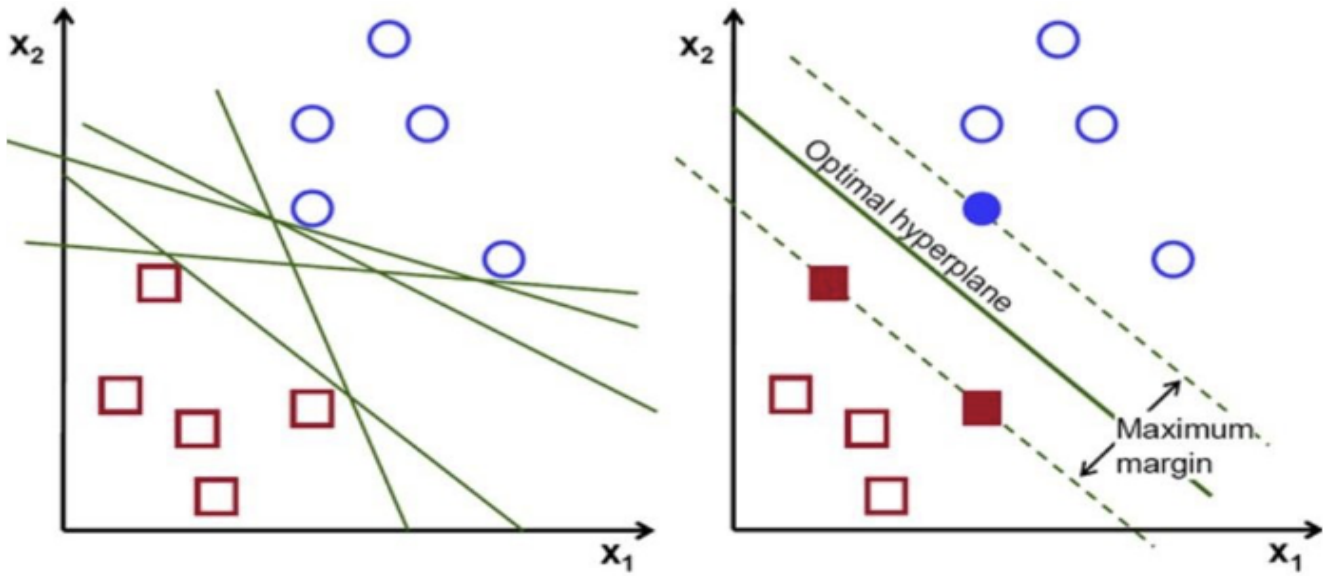


Figure 11. SVM-Finding the optimal hyper line

A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel(3.9) can distinguish curved or nonlinear input space.

$$K(x, x_i) = 1 + \text{sum}(x * x_i)^d \quad (3.9)$$

where, d is the degree of the polynomial.

3.3.5 Random Forest

Random Forest is an ensemble consisting of a bagging of unpruned decision trees with a randomized selection of features at each split. Each individual tree in the random forest produces a prediction and the prediction with the most votes is the final prediction. According to the No Free Lunch theorem: There is no algorithm that is always the most accurate, thus RF is more accurate and robust than the individual classifiers.

The random forest algorithm can be expressed as

$$F(x) = \arg \max_l \left\{ \sum_{i=1}^z T(A(B, \theta_k)) \right\} \quad (3.10)$$

where $F(x)$ is the random forest model, j is the target category variable and F is the characteristic function.

Random Forest Algorithm

- 1: Draw Bootstrap sample sets m_{sub} with replacement
- 2: Choose a sample set as the root node and train in a completely split way
- 3: Select f_{sub} randomly from f and choose the best feature to split the node by using the minimum principle of Gini impurities .
- 4: Let the nodes grow to the maximum extent. Label the nodes with a minimum impurity as leaf nodes.
- 5: Repeat steps 2-4 until all nodes have been trained or labeled as leaf nodes.
- 6: Repeat steps 2-5 until all CART has been trained
- 7: Output the random forest with m_{sub} CART trees

3.4 Deep Learning Models

3.4.1 CNN

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate(as shown in Fig.12) one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

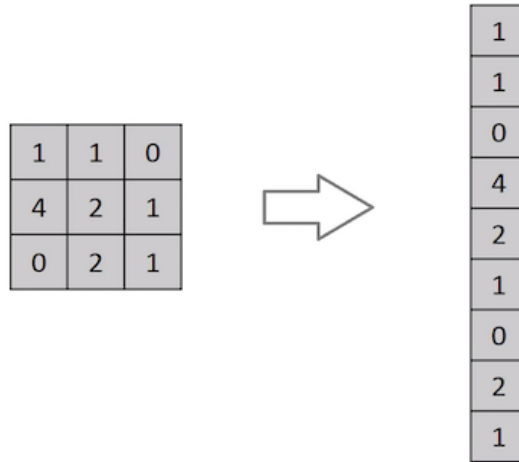


Figure 12. Matrix of pixel value

3.4.2 LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more.

LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field. There are few that are better at clearly and precisely articulating both the promise of LSTMs and how they work than the experts that developed them. make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things.

Long-term state $c(t-1)$ first processed by the forget gate, dropping some memories, and plus some new memories selected by the input gate. Then $c(t-1)$ becomes the result $c(t)$. The long-term state $c(t-1)$ is first processed by the forget gate, dropping some memories, and plus some new memories selected by the input gate. Then $c(t-1)$ becomes the result $c(t)$ (as shown in Fig.13).

Besides that, $c(t-1)$ is copied and passed through the tanh function and filtered by the output gate to compute the short-term state $h(t)$ which is also the cell's output at t time step, y_t . Normally in a basic RNN cell, there is only one fully connected layer that outputs g_t . However in the LSTM cell, there are three more gate controllers layers. Due to logistic activation function, their computation results range from 0 to 1. By using element-wise multiplication, if they output zeros, the according gate will be

closed and if they output ones the gate will be open. The forget gate determines what to delete in the long-term. The input gate determines what should be added to the long-term state. The output gate determines what parts of the long-term state should be read and output at the current time step.

The involved computation in LSTM cell can be summarized as follow:

$$\begin{aligned}
 i_{(t)} &= \sigma(W_{xi}^T x_{(t)} + W_{hi}^T h_{(t-1)} + b_i) \\
 f_{(t)} &= \sigma(W_{xf}^T x_{(t)} + W_{hf}^T h_{(t-1)} + b_f) \\
 o_{(t)} &= \sigma(W_{xo}^T x_{(t)} + W_{ho}^T h_{(t-1)} + b_o) \\
 g_{(t)} &= \tanh(W_{xg}^T x_{(t)} + W_{hg}^T h_{(t-1)} + b_g) \\
 c_{(t)} &= f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \\
 y_{(t)} &= h_t = o_{(t)} \otimes \tanh(c_{(t)})
 \end{aligned}$$

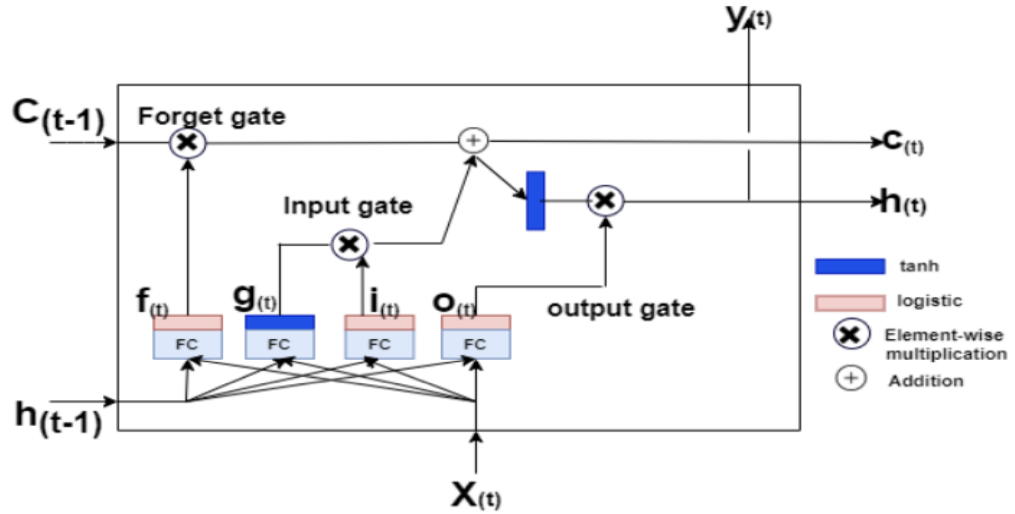


Figure 13. LSTM

3.4.3 Gated Recurrent Unit

GRUs are an improved version of standard recurrent neural network. To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate (as shown in Fig. 14). Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or removing information which is irrelevant to the prediction.

In GRU cells, the long short term states are merged into a single vector $h(t)$. And there are only two gate controllers $z(t)$ and $r(t)$. When $z(t)$ outputs 1, the forget gate is open and the input gate is closed. when $z(t)$ outputs 0, it will close the forget gate and open the input gate.

GRU cell [9] is a simple version of LSTM cell, but sometimes its performance is even better than LSTM(as shown in Fig.15). This conclusion can also be proven in our experiments which shows that their performance is almost the same but the time GRU neural network spent on training is much shorter. In GRU cells, the long short term states are merged into a single vector $h(t)$. And there are only two gate controllers $z(t)$ and $r(t)$. When $z(t)$ outputs 1, the forget gate is open and the input gate is closed. when $z(t)$ outputs 0, it will close the forget gate and open the input gate. At every time step, there always will be an output unlike in LSTM cells where there exists an output gate. And $r(t)$ controls what content in the previous state should be sent to the main layer $g(t)$.

$$z(t) = \sigma(W_{xz}^T x(t) + W_{hz}^T h(t-1) + b_z)$$

$$r(t) = \sigma(W_{xr}^T x(t) + W_{hr}^T h(t-1) + b_r)$$

$$g(t) = \tanh(W_{xg}^T x(t) + W_{hg}^T (r(t) \otimes h(t-1)) + b_g)$$

$$h(t) = z(t) \otimes h(t-1) + (1 - z(t)) \otimes g(t)$$

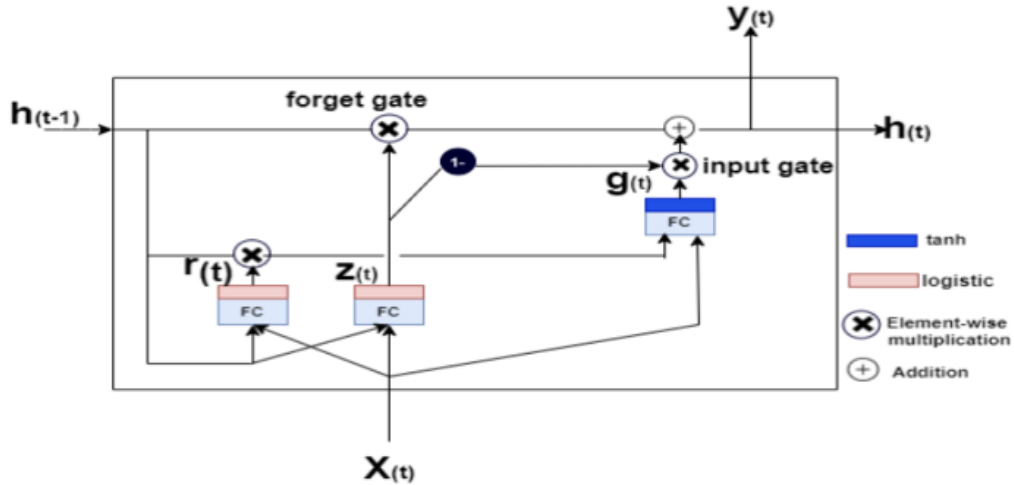


Figure 14. GRU

GRUs are very similar to Long Short Term Memory(LSTM). Just like LSTM, GRU uses gates to control the flow of information. They are relatively new as compared to LSTM. This is the reason they offer some improvement over LSTM and have simpler architecture.

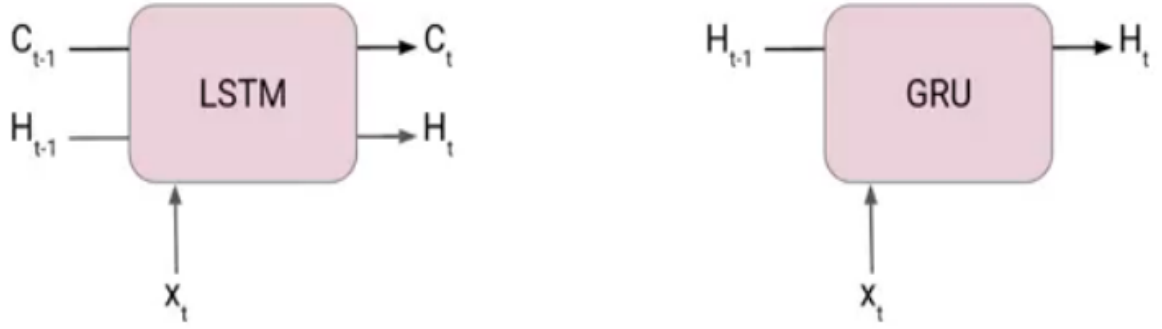


Figure 15. Comparison between LSTM & GRU

3.5 Hold Out Cross Validation Method

For best model selection, a hold out cross-validation method is evaluated. In the hold out method, the data set is divided into two parts for training and testing. In the experiments, 80% of the instances are used to train the classifiers and the remaining 20% of the datasets are used for testing the classifiers.

3.6 Model Evaluation Criteria

The employed accuracy, precision, recall, and f1-score for model evaluation. These evaluation metrics are expressed as follows:

$$Accuracy (Acc) = \frac{TP + TN}{TP + TN + FP + FN} 100\%$$

$$Recall (Re) = \frac{TP}{TP + FN} 100\%$$

$$Precision (Pre) = \frac{TN}{TN + FP} 100\%$$

$$F1 - score = 2 \frac{(precision)(recall)}{precision + recall}$$

$$Accuracy (Acc) = (TP + TN) / (TP + TN + FP + FN) * 100\%$$

$$Recall (Re) = TP / (TP + FN) * 100\%$$

$$Precision (Pre) = TN / (TN + FP) * 100\%$$

$$F1 - score = 2 * (precision) * (recall) / (precision + recall)$$

CHAPTER 4

RESULTS

Data Cleaning

Preprocessing Text to get Stemmed and Lemmatized Corpus

```
In [14]: porter_stemmer = PorterStemmer()
         lemmatizer = WordNetLemmatizer()
```

```
In [15]: # nltk.download('all')
```

```
In [16]: stemmed_text = []
         lemmatized_text = []
         final_text_result = []
         for text in dataset['final_text']:
             result = re.sub('[^a-zA-Z]', ' ', text)
             result = result.lower()
             result = result.split()
             result = [r for r in result if r not in set(stopwords.words('english'))]
             final_text_result.append(" ".join(result))
             stemmed_result = [porter_stemmer.stem(r) for r in result]
             stemmed_text.append(" ".join(stemmed_result))
             lemmatized_result = [lemmatizer.lemmatize(r) for r in result]
             lemmatized_text.append(" ".join(lemmatized_result))
```

```
In [17]: print(len(final_text_result))
         print(len(stemmed_text))
         print(len(lemmatized_text))
```

```
44898
44898
44898
```

Create models using CountVectorizer and TFIDF Vectorizer with stemmed text and lemmatized text ¶

```
In [20]: def get_prediction(vectorizer, classifier, X_train, X_test, y_train, y_test):
         pipe = Pipeline([('vector', vectorizer),
                           ('model', classifier)])
         model = pipe.fit(X_train, y_train)
         y_pred = model.predict(X_test)
         print("Accuracy: {}".format(round(accuracy_score(y_test, y_pred)*100,2)))
         cm = confusion_matrix(y_test, y_pred)
         print("Confusion Matrix: \n", cm)
         print("Classification Report: \n", classification_report(y_test, y_pred))
```

```
In [21]: print("*****USING STEMMED TEXT*****")
         X_train, X_test, y_train, y_test = train_test_split(stemmed_text, dataset['Category'], test_size = 0.3, random_state=0)
         classifiers = [LogisticRegression(), SGDClassifier(), MultinomialNB(), BernoulliNB(), LinearSVC(),
                        KNeighborsClassifier(n_neighbors=5), DecisionTreeClassifier(), GradientBoostingClassifier(),
                        RandomForestClassifier(), XGBClassifier()]
         for classifier in classifiers:
             print("\n\n", classifier)
             print("*****Usng Count Vectorizer*****")
             get_prediction(CountVectorizer(), classifier, X_train, X_test, y_train, y_test)
             print("*****Usng TFIDF Vectorizer*****")
             get_prediction(TfidfVectorizer(), classifier, X_train, X_test, y_train, y_test)
```

The results obtained using countvectorizer & TF-IDF with stemmed text are (mentioned in Table.2):

Table 2: Stemmed Text results

	Using count vectorizer	Using TF-IDF
LR	99.63	98.66
NB	95.3	93.56
KNeighbours	78.25	86.82
Decision Tree	99.7	99.62
Random Forest	99.05	98.9

```
In [22]: print("*****USING LEMMATIZED TEXT*****")
X_train, X_test, y_train, y_test = train_test_split(lemmatized_text, dataset['Category'], test_size = 0.3, random_state
classifiers = [LogisticRegression(), SGDClassifier(), MultinomialNB(), BernoulliNB(), LinearSVC(),
                KNeighborsClassifier(n_neighbors=5), DecisionTreeClassifier(), GradientBoostingClassifier(),
                RandomForestClassifier(), XGBClassifier()]
for classifier in classifiers:
    print("\n\n", classifier)
    print("*****Usng Count Vectorizer*****")
    get_prediction(CountVectorizer(), classifier, X_train, X_test, y_train, y_test)
    print("*****Usng TFIDF Vectorizer*****")
    get_prediction(TfidfVectorizer(), classifier, X_train, X_test, y_train, y_test)
```

The results obtained using countvectorizer & TF-IDF with lemmatized text are (mentioned in Table.3):

Table 3: Lemmatized Text results

	Using count vectorizer	Using TF-IDF
LR	99.65	98.73
NB	95.46	93.87
KNeighbours	77.82	87.5
Decision Tree	99.65	99.62
Random Forest	99.25	98.11

Using One hot representation using Lemmatized Text

```
In [ ]: voc_size = 5000
onehot_lemmatized_text = [one_hot(word, voc_size) for word in lemmatized_text]
print(len(onehot_lemmatized_text))
onehot_lemmatized_text[0]

In [ ]: sent_length = 400
embedded_text = pad_sequences(onehot_lemmatized_text, padding='pre', maxlen=sent_length)
print(embedded_text)

In [ ]: embedding_vector_features = 600
model2 = Sequential()
model2.add(Embedding(voc_size, embedding_vector_features, input_length=sent_length))
model2.add(LSTM(100))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(loss='binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model2.summary()

In [ ]: X_final = np.array(embedded_text)
y_final = dataset['Category']

In [ ]: X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size = 0.3, random_state = 0)

In [ ]: model2.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20, batch_size = 256, callbacks=([reduce_lr, earl
```

Evaluate model

```
In [42]: y_pred = model2.predict_classes(X_test)
print("Accuracy: {}".format(round(accuracy_score(y_test, y_pred)*100,2)))
```

Accuracy: 98.4

```
In [43]: cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix: \n", cm)
```

Confusion Matrix:
[[6966 103]
[113 6288]]

```
In [44]: print("Classification Report: \n", classification_report(y_test, y_pred))
```

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	7069
1	0.98	0.98	0.98	6401
accuracy			0.98	13470
macro avg	0.98	0.98	0.98	13470
weighted avg	0.98	0.98	0.98	13470

With One hot vector and stemmed text LSTM model is giving 98.7 % accuracy.

Gated Recurrent Net

GRU

```
In [119]: emb_dim = embedding_matrix.shape[1]
gru_model = Sequential()
gru_model.add(Embedding(vocab_len, emb_dim, trainable = False, weights=[embedding_matrix]))
gru_model.add(GRU(128, return_sequences=False))
gru_model.add(Dropout(0.5))
gru_model.add(Dense(1, activation = 'sigmoid'))
gru_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(gru_model.summary())
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 100)	1000000
gru (GRU)	(None, 128)	88320
dropout (Dropout)	(None, 128)	0
dense (Dense)	(None, 1)	129
Total params: 1,088,449		
Trainable params: 88,449		
Non-trainable params: 1,000,000		
None		

```
In [141]: train_gru_results = gru_model.evaluate(X_train_pad, y_train, verbose=0, batch_size=256)
test_gru_results = gru_model.evaluate(X_test_pad, y_test, verbose=0, batch_size=256)
print("Train accuracy: {}".format(train_gru_results[1]*100))
print("Test accuracy: {}".format(test_gru_results[1]*100))
```

Train accuracy: 99.99363422393799
Test accuracy: 99.84409809112549

```
In [142]: y_pred = gru_model.predict_classes(X_test_pad)
print("Accuracy: {}".format(round(accuracy_score(y_test, y_pred)*100,2)))
```

Accuracy: 99.84

With GRU , the accuracy is highest 99.84%

LSTM

```
In [59]: lstm_model = Sequential()
lstm_model.add(Embedding(vocab_len, emb_dim, trainable = False, weights=[embedding_matrix]))
lstm_model.add(LSTM(128, return_sequences=False))
# lstm_model.add(Dropout(0.25))
lstm_model.add(Dense(1, activation = 'sigmoid'))
lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(lstm_model.summary())
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
=====		
embedding_3 (Embedding)	(None, None, 100)	1000000

lstm_3 (LSTM)	(None, 128)	117248

dense_3 (Dense)	(None, 1)	129
=====		
Total params: 1,117,377		
Trainable params: 117,377		
Non-trainable params: 1,000,000		

None		

Epochs

Epochs

```
In [62]: train_lstm_results = lstm_model.evaluate(X_train_pad, y_train, verbose=0, batch_size=256)
test_lstm_results = lstm_model.evaluate(X_test_pad, y_test, verbose=0, batch_size=256)
print("Train accuracy: {}".format(train_lstm_results[1]*100))
print("Test accuracy: {}".format(test_lstm_results[1]*100))
```

Train accuracy: 99.87590909004211
Test accuracy: 99.73273873329163

```
In [63]: y_pred = lstm_model.predict_classes(X_test_pad)
print("Accuracy: {}".format(round(accuracy_score(y_test, y_pred)*100,2)))
```

Accuracy: 99.73

CHAPTER 5

CONCLUSION AND FUTURE WORK

In Conclusion, five machine learning models and three deep learning models are evaluated on two fake news datasets of different sizes in terms of accuracy, precision, recall, F1- score. According to some experiments, some models like K-Nearest Neighbors had better performance on small datasets and other models like Decision Tree, Support Vector Machine, Logistic Regression, CNN, GRU, LSTM had a much worse performance on small datasets. To select the best model, a corrected version of McNemar's test was used to determine if models' performance is significantly different. According to the final experiments, among all individual models, Random Forest with TF-IDF has the highest accuracy on the ISOT dataset and Logistic Regression with TF-IDF has the highest accuracy on the KDnugget dataset.

The experimental results of these two best models demonstrated that the proposed stacking method achieved 99.94% accuracy on the ISOT dataset was very high as compared to individual models. The compared results are used to other existing work and conclude that our stacking model is much better. Due to the high performance of our proposed stacking methods, it is recommended for the detection of fake news.

The major innovation of this research work is as follows: Firstly, five machine learning and three deep learning models have been trained in order to compare the performance difference between machine learning and deep learning models. Secondly, datasets of different sizes are used for evaluation of the proposed method to test models' robustness on datasets of different sizes. Thirdly, a corrected version of McNemar's statistic was employed to test to decide if there really are significant differences between two model's performance and determine the best individual model for fake news detection. Lastly, the proposed stacking model to improve the individual model performance.

The proposed method for fake news detection has been compared with state of the art methods in Table. According to the Table, in terms of accuracy this method achieved 99.94% accuracy on the ISOT dataset which is so much better than existing methods. Therefore, the proposed method is highly recommended for detection of fake news and it could be easily employed in a real environment.

Model's accuracy is a common metric that researchers used to select the models. There are a lot of models that had very similar accuracy. Thus, employed a corrected version of McNemar's test to determine if models' performance has significant difference in accuracy.

Future Work:

In future, more experiments on other data sets are going to be performed in different languages. Also trying to use more different machine learning and deep learning models for fake news detection. Also collecting more fake and real news data in different languages to detect fake news in different countries.

CHAPTER 6

REFERENCES

- [1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [2] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
- [3] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.
- [4] H.-L. Chen, B. Yang, J. Liu, and D.-Y. Liu, “A support vector machine classifier with rough set-based feature selection for breast cancer diagnosis,” *Expert Syst. Appl.*, vol. 38, no. 7, pp. 9014–9022, Jul. 2011.
- [5] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” 2014, arXiv:1406.1078. [Online].
- [6] Y. Long, “Fake news detection through multi-perspective speaker profiles, version I17-2043,” *Assoc. Comput. Linguistics*, Stroudsburg, PA, USA, Tech. Rep., Nov. 2017, vol. 2. [Online].
- [7] S. D. Bhattacharjee, A. Talukder, and B. V. Balantrapu, “Active learning based news veracity detection with feature weighting and deep-shallow fusion,” in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 556–565.
- [8] S. Gilda, “Evaluating machine learning algorithms for fake news detection,” in *Proc. IEEE 15th Student Conf. Res. Develop. (SCOREd)*, Dec. 2017, pp. 110.
- [9] G. E. R. Agudelo, O. J. S. Parra, and J. B. Velandia, “Raising a model for fake news detection using machine learning in Python,” in *Proc. Conf. eBus., e-Services e-Soc*. Cham, Switzerland: Springer, 2018, pp. 596–604.

- [10] H. Ahmed, I. Traore, and S. Saad, “Detecting opinion spams and fake news using text classification,” *Secur. Privacy*, vol. 1, no. 1, p. e9, Jan. 2018.
- [11] M. L. Della Vedova, E. Tacchini, S. Moret, G. Ballarin, M. DiPierro, and L. de Alfaro, “Automatic online fake news detection combining content and social signals,” in *Proc. 22nd Conf. Open Innov. Assoc. (FRUCT)*, May 2018, pp. 272–279.
- [12] O. Ajao, D. Bhowmik, and S. Zargari, “Fake news identification on Twitter with hybrid CNN and RNN models,” in *Proc. 9th Int. Conf. Social Media Soc.*, Jul. 2018, pp. 226–230.
- [13] A. U. Haq, J. Li, M. H. Memon, J. Khan, S. U. Din, I. Ahad, R. Sun, and Z. Lai, “Comparative analysis of the classification performance of machine learning classifiers and deep neural network classifiers for prediction of Parkinson disease,” in *Proc. 15th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. (ICCWA TIP)*, Dec. 2018, pp. 101–106.
- [14] A. U. Haq, J. P. Li, M. H. Memon, J. Khan, A. Malik, T. Ahmad, A. Ali, S. Nazir, I. Ahad, and M. Shahid, “Feature selection based on L1-norm support vector machine and effective recognition system for Parkinson’s disease using voice recordings,” *IEEE Access*, pp. 37718–37734, 2019.
- [15] A. Jain, A. Shakya, H. Khatter, and A. K. Gupta, “A smart system for fake news detection using machine learning,” in *Proc. Int. Conf. Issues Challenges Intel. Comput. Techn. (ICICT)*, vol. 1, Sep. 2019, pp. 1–4.
- [16] R. K. Kaliyar, A. Goswami, and P. Narang, “Multiclass fake news detection using ensemble machine learning,” in *Proc. IEEE 9th Int. Conf. Adv. Comput. (IACC)*, Dec. 2019, pp. 103–107.
- [17] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, “MIMETIC: Mobile encrypted traffic classification using multimodal deep learning,” *Comput. Netw.*, vol. 165, Dec. 2019, Art. no. 106944.
- [18] M. Amjad, G. Sidorov, A. Zhila, H. Gómez-Adorno, I. Voronkov, and A. Gelbukh, “‘Bend the truth’: Benchmark dataset for fake news detection in urdu language and its evaluation,” *J. Intell. Fuzzy Syst.*, vol. 39, no. 2, pp. 2457–2469, 2020.

- [19] A. L. Edwards, “Note on the ‘correction for continuity’ in testing the significance of the difference between correlated proportions,” *Psychometrika*, vol. 13, no. 3, pp. 185–187,
- [20] M. H. Goldani, S. Momtazi, and R. Safabakhsh, “Detecting fake news with capsule neural networks,” 2020, arXiv:2002.01030. [Online].
- [21] A. U. Haq, J. Li, M. H. Memon, J. Khan, and S. U. Din, “A novel integrated diagnosis method for breast cancer detection,” *J. Intell. Fuzzy Syst.*, vol. 38, no. 2, pp. 2383–2398, Feb. 2020.
- [22] S. Kula, M. Choraś, R. Kozik, P. Ksieniewicz, and M. Woźniak, “Sentiment analysis for fake news detection by means of neural networks,” in *Proc. Int. Conf. Comput. Sci. Cham, Switzerland: Springer*, 2020, pp. 653–666.
- [23] S. Kumar, R. Asthana, S. Upadhyay, N. Upreti, and M. Akbar, “Fake news detection using deep learning models: A novel approach,” *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, p. e3767, Feb. 2020.
- [24] R. K. Kaliyar, A. Goswami, P. Narang, and S. Sinha, “FNDNet—A deep convolutional neural network for fake news detection,” *Cognit. Syst. Res.*, vol. 61, pp. 32–44, Jun. 2020.
- [25] Z Khanam¹, B N Alwasel, H Sirafi and M Rashid, “Fake News Detection Using Machine Learning Approaches” published under license of IOP publications., vol. 1099, Aug 2021.