Northumbria University
NEWCASTLE

Dissertation

KF7029
MSc Computer Science and Digital Technologies Project

Student Name: Konark Karna

Supervisor Name: Dr Alan Godfrey

Second Marker Name: Prof Rebecca Strachan

Dissertation Title: Comparative Analysis of Supervised Learning Methods for Fake News Detection

2020/2021

Declaration

I declare the following:

(1) that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to **ALL** sources be they printed, electronic or personal.

(2) the Word Count of this Dissertation is 12292.

(3) that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on the eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other Department or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

(5) I have read the Northumbria University Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

SIGNED: Konark Karna

DATE: 27-05-2021

Abstract

Fake news detection is a challenging problem, and there are many different artificial intelligence approaches developed to detect fake news. However, it is still unclear which approach is better and where further research should lead. To accomplish this goal, we conducted this study to compare the existing supervised learning methods for classifying fake news. We conducted our study on 'Liar, Liar, Pants on Fire' dataset with 12.8K columns. We implemented NLP technique TF-IDF and evaluated the performance of seven machine learning algorithms i.e., Decision Tree, Random Forest, Naïve Bayes, XGBoost, Logistic Regression, Support Vector Machine (SVM) and LinearSVC and six deep learning methods namely Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network, Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU) and Bi-Directional LSTM. Our results reveal that machine learning algorithms like Random Forest, Logistic Regression and SVM and deep learning methods like LSTM and GRU are the most efficient for fake news detection task, whereas Decision Tree, LinearSVC, Artificial Neural Network, Recurrent Neural Network are least efficient. We have also presented evaluation of our implementation and model performance. Finally, we discussed how the fake news detection problem can be approached in future for better results.

Contents

1 Introduction

Fake news has its historical predecessor in the concept of 'Propaganda', and just like propaganda, fake news comprises lies, deceit, and misinformation masquerading as truthful information. In present times, with the invention and widespread use of the internet, it has now become a major tool to manipulate public opinion and decision-making. Fake news creators create it with intriguing titles, photoshopped images, and sentiment provoking words. People tend to read and share such stimulating news among their peers, and consequently making the problem even worse.

Fake news nowadays has started to impact our society on many aspects, be it on the political front or economical front. It is such a problematic issue now that 'fake news' got selected as the "word of the year" in the year 2016 by Macquarie Dictionary (Bolton and Yaxley, 2017), and recently even the World Economic Forum decided to warn people about it (World Economic Forum, 2020).

Challenge what fake news poses has been studied by many researchers, and in the field of artificial intelligence many researchers are working to solve this problem. In this study, we are evaluating currently existing supervised learning methods of artificial intelligence to detect fake news. There are already a multitude of methods presented in literature and we aim to compare those methods. This comparative study will help adjudge the efficient approach that would further help in research work. Seven machine learning algorithms and six deep learning methods have been chosen to compare and analyse in this study. We have selected Decision Tree, Random Forest, XGBoost, Naïve Bayes, Logistic Regression, Support Vector Machine, Linear Support Vector Classifier, Artificial Neural Network, Convolutional Neural Network, Recurrent Neural Network, Long Short-Term Memory, Gated Recurrent Units, and Bi-Directional LSTM models to evaluate on their performance of fake news detection task.

This study is organised as follows: Chapter 2 contains review of research work on the issue of fake news detection. Chapter 3 contains our research approach. We are discussing algorithms and methods to be implemented for this study. Chapter 4 covers our main research work and gathering results and comparison. Chapter 5

contains evaluation of our fake news detection models and reflection on our work. The final chapter 6 contains our suggestions for further research on fake news detection problem.

1.1 Background

In this information age, people extensively rely on the internet for almost every decision making. Online news, articles, statistics, and reviews etc deeply influence people's perception and rationale behind each decision making of their lives. Fake news, on the internet, is created with the sheer intent of intercepting this decision-making process. It is created with sensationalist, flashy headlines to attract the reader's attention, and contains misleading information and pictures in the content. We can, arguably, say that considering the amount of time people spend on the internet reading and sharing information, everyone is equally susceptible to fake news.

Fake news is a type of yellow journalism or propaganda (Ghafari, Yakhchi, Beheshti and Orgun, 2018), and many political groups and business organisations are now using fake news to suit their political or financial gain. They are even opting to generate fake news through bots, and trolls, and paid platforms are used to spread it to greater number of people (Zannettou, Sirivianos, Blackburn and Kourtellis, 2019).

Fake News now poses a major threat to political stability and business sector. On the political front, people can be misled into re-establishing their decisions by fake news. For instance, a fake news circulating during European Union (EU) referendum that the United Kingdom (UK) sent £350 million per week to the EU which can be spent on National Health Services (NHS) misled many people into Brexit (Rose, 2017). Similarly, fake news misled people during US 2016 elections too. (Allcott and Gentzkow, 2017). On the business front, fake news, and reviews can severely impact brand value of products and services (Mills and Robson, 2019). For instance, once Apple shares fell 10% in 10 minutes when a fake news of Steve Jobs heart-attack was posted online (Hargreaves, 2020). Fake news can hamper decision making of public on other fronts as well, such as climate change. Fake news is making people believe that it is a hoax (Van der Linden et al, 2017), and even in the pandemic days of Covid-19, fake news made people believe that virus is bioengineered by China (Pennycook

et al., 2020) and will not survive in the heat of 26-27ºC (Brennen, Simon, Howard, and Nielsen, 2020).

We could say that the wellbeing of a society, economic progress and democratic functioning of any country is dependent on their well-informed population. Well-informed civilians could make rational decisions for themselves and for the greater good of their society (Lewandowsky et al., 2012). However, fake news creates opposite effects. It hampers both economic progress and democratic functioning of society by misinforming general people and making them stay in their echo chamber with biased cluster of information (Bakir and McStay, 2017)

This problem of fake news is currently so widespread on the internet that it is estimated by the year 2022, people will be consuming more fake news than real news. (Gartner, 2020). So, for the betterment of everyone, and healthy society, now it has become an important issue to identify fake news. Recently, even inventor of the internet Tim-Berners Lee stated that fake news is one of the disturbing things on the internet and 'spread like wildfire' (Swartz, 2020).

1.2 Motivation

Many organisations and researchers are involved in solving this problem of fake news identification, and many different approaches conceptualized till now. One of the best approaches to identify fake news is to cross-verifying it with the experts. Certain organisations like Snopes (Snopes.com, 2020), FactCheck (FactCheck.org, 2020), and PolitiFact (Politifact.com, 2020) are implementing it. They are rating the accuracy of news by chosen political analysts. However, this approach is time-consuming, and can only be applied on a certain amount of news each day.

In the field of artificial intelligence, Google has started an initiative named Factmata(Factmata.com, 2020) and an automated fact-checker called ContentCheck (ContentCheck, 2020) is also developed. Along with these, many individual researchers are working on the problem of fake news identification, and many brilliant solutions have been proposed. We have, now, a plethora of existing research based on machine learning and deep learning algorithms implemented with natural language processing method to accurately classify fake news. However, most of the research has been conducted on different datasets and modelled on single

algorithms. We find it paramount to conduct a comparative analysis of existing supervised learning methods utilized for fake news detection on a single dataset. It will present a realistic understanding of accuracy a model can predict in comparison with other models.

1.3 Aims

To study the existing methods of artificial intelligence for identifying fake news and present a comparative analysis of existing supervised learning methods for fake news detection.

1.4 Objectives

To conduct this study of comparative analysis of supervised learning methods, many sub-goals or objectives had to be achieved:

- Extensive research work had to be identified and evaluated for the existing methods of fake news detection in the field of artificial intelligence.
- Supervised learning methods had to be identified suitable for fake news detection.
- Research work on machine learning, deep learning and natural language processing concepts had to be extensively studied to develop deeper understanding of research studies and conduct our own.
- Programming language, libraries, and toolkits like Python, SciKit-Learn, Tensorflow, Keras, and NLTK had to be learnt to implement while conducting our analysis.
- A labelled dataset, ideally with more than 10,000 size, must be collected so that model implemented would not suffer the problem of undersampling and results could be adjudged as reliable.
- Develop individual models based on identified methods for fake news detection.
- Compare fake news detection models based on the performance metrics obtained through our research.

1.5 Structure of the Report

Structure of the report is presented as follows: Chapter 2 has a review of all the existing methods for fake news detection and comparative studies. Methods to be implemented in our research are identified. Then chapter 3, presents our understanding of machine learning algorithms, deep learning methods and natural language processing concepts which we will approach in our research. In chapter 4, we are presenting our research work and collecting results. We are also comparing our results on performance metrics. Then, in the next chapter 5, we will evaluate our model performances and reflect on our comparative analysis. Lastly, in chapter 6, we will present our suggestions for future work in this field of fake news detection.

2 Related Work

In this chapter, we will review existing approaches of artificial intelligence developed for fake news detection problem. We will start from different types of fake news detection techniques. Next, we will understand existing machine learning models for fake news detection, and lastly, we will cover existing comparative analysis work on supervised learning methods.

In the field of artificial intelligence, fake news detection approaches can be classified into two ways: based on context of news and based on content of news.

Context-based approach

Context of the news provides information on author, publication time, page views, comments, popularity, etc. It is, therefore, more suitable for detecting fake news or rumours on social media where information on number of retweets, mentions, likes, comments, transmission rate and degree of popularity are available. Using context-based approach (Shu, Wang and Liu, 2019) proposed a tri-relationship embedding framework, Tri-Fake News (TriFN) model, investigating tri-relationship between publishers, news pieces, and users. (Tschiatschek et al., 2018) utilized crowds flagging accuracy to detect fake news. In another work, a model has been proposed to detect rumour over a varying time window i.e., three days to two months (Kwon, Cha and Jung, 2017). Another model proposed a back-tracking method to track sources of information and detected fake news with 80% accuracy (Ko et al., 2019).

Content -based approach

Content-based approach for fake news detection can be implemented using both images, and text of the news articles. (Singhal et al., 2019) used both features using ImageNet and textual features to predict fake news with the highest 89% accuracy. Similarly, (Yang et al., 2018) proposed a solution that has precision of 92% using textual and image latent features modelled on convolution neural network. Another model, named Multimodal Variational Autoencoder (MVAE) used text and image features with autoencoder to score accuracy of 82% (Khattar, Goud, Gupta and Varma, 2019).

However, most of the existing models for identifying fake news use text data of news articles for implementing suitable machine learning and deep learning methods. Proposed fake news detection models vary in their complexity a lot. (Granik and Mesyura, 2017) presented one of the simplest models using Naïve Bayes classifier which achieved a moderate accuracy of 74%. Another simple model (Bhutani, Rastogi, Sehgal and Purwar, 2019) presented the idea of adding sentiment as a column in the dataset and scored accuracy of 84% and 83% using Naïve Bayes and Random Forest algorithms. (Ahmed, Traore and Saad, 2017) used n-gram from n=1 to n=4 and Linear Support Vector Machine (LSVM) to acquire an accuracy of 92%. Another model used LSVM with Linguistic Inquiry and Word Count (LIWC) to extract features and scored an accuracy of 91% (Perez, Kleinberg and Lefevre, 2017). An analytical model on sentence type is also proposed using K-Means algorithm and achieved 92% accuracy (Zhang et al., 2019). Another model utilized credibility of news articles, subject, and creator in a diffusive neural network model to predict with 63% accuracy (Zhang, Dong and Philip, 2019).

Word embedding techniques like Global Vector for Word Representation (GloVe) and Word2Vec of Natural Language Processing (NLP) are also used on text data in a few proposed models. (Popat, Mukherjee, Yates and Weikum, 2018) used Glove with Convolution Neural Network (CNN) to get an accuracy of 69%. Similarly, (Kaliyar, Goswami, Narang and Sinha, 2020) also used GloVe word embedding technique but, with deep convolutional neural network (CNN) and achieved an accuracy of 98%. Massive variance in results with similar algorithms can be accredited to the dataset used in respective models. Another word embedding technique named Word2Vec is also employed with CNN in a few models for fake news detection. (Jang, Kim and Kim, 2019) used Continuous Bag-of-Word of Word2Vec with CNN and scored highest accuracy of 93%. (Fang et al., 2019) employed an improved model of attention mechanism of natural language processing - self multi head attention mechanism on CNN and achieved an accuracy of 95.9%. (Kaliyar, Goswami and Narang, 2020) used multi-channel deep convolutional neural network (CNN) on FN-COV dataset of circulating real-world fake news on covid-19 pandemic and achieved an accuracy of 98%.

Recurrent Neural Network (RNN) is also implemented in many presented systems. (Bahad, Saxena and Kamal, 2019) used Long Short-Term Memory (LSTM) architecture of RNN to score accuracies of 91% and 98% on two different datasets with GloVe NLP technique. Similarly, (Barua et al., 2019) employed Gated Recurrent Units (GRU) of RNN to score an accuracy of 82%. (Jadhav and Thepade, 2019) used Deep Semantic Structural Model (DSSM) with LSTM of RNN to get the highest accuracy of 99% when the dataset is split on a 75-25 basis between training set and test set. (Roy, Basek, Ekbal and Bhattacharyya, 2018) used Bi-directional LSTM combined with CNN to get an accuracy of 55%. Bi-LSTM again used in another model (Kim and Jeong, 2019) with attention mechanism to get an accuracy of 88% on true, and 47% on false news.

Few researchers have also presented comparative study of supervised learning methods. (Reis et al., 2019) compared K-Nearest Neighbours (KNN), Naïve Bayes (NB), Random Forest (RB), Support Vector Machine (SVM) and XGBoost on Buzzfeed dataset of 2286 articles and found XGBoost to be the most effective. (Agarwal, Sultana, Malhotra and Sarkar, 2019) compared NB, Logistic Regression (LR), SVM, and RF on "Liar, Liar, Pants on Fire" dataset and adjudged all algorithms to be more or less equivalent with accuracy close to 62%. (Poddar, Amali D. and Umadevi, 2019) compared Naïve Bayes, Logistic Regression, Decision Tree, Support Vector Machine, and Artificial Neural Network on a dataset collected from Kaggle with TF-IDF implementation on text and estimated SVM to be a better choice. Two more comparative studies have found SVM to be the most effective for the fake news detection problem. (Gravanis, Vakali, Diamantaras and Karadais, 2019) implementing Word2Vec when compared with K-NN, Decision Tree, NB, AdaBoost and Bagging on UNBiased Fake News Dataset, and (Faustini and Covões, 2020) when compared with KNN, NB, and Random Forest on a merged dataset of five different collections. (Kesarwani, Chauhan, Nair and Verma, 2020) compared Support Vector Machine, Random Forest, Logistic Regression, Naïve Bayes and K-NN on Kaggle dataset and found Logistic Regression to be most effective with 98.5% accuracy. (Kaur, Kumar and Kumaraguru, 2019) used Kaggle, Reuters and News Trends dataset with twelve different algorithms and found Logistic Regression and LinearSVC to be

Comparative Study	Algorithms	Dataset	Results
(Reis et al., 2019)	K-NN, Naïve bayes, Random Forest, SVM, XGBoost	Buzzfeed	XGBoost
(Agarwal, Sultana, Malhotra, and Sarkar, 2019)	Naïve Bayes, SVM, Logistic Regression, Random Forest	"Liar, Liar, Pants on Fire"	All equal with ~62%
(Poddar, Amali D, and Umadevi, 2019)	Naïve Bayes, SVM, Logistic Regression, Decision Tree, Artificial Neural Network	Kaggle	SVM;92% Accuracy
(Gravanis, Vakali, Diamantaras, and Karadais, 2019)	K-NN, Decision Tree, Naïve Bayes, SVM, AdaBoost, Bagging	UNBiased	SVM; 95% Accuracy
(Faustini and Covoes, 2020)	K-NN, Naïve Bayes, Random Forest, SVM	Five different datasets	SVM; 95% highest Accuracy
(Kesarwani, Chauhan, Nair and Verma, 2020)	SVM, Random Forest, Logistic Regression, Naïve Bayes, K-NN	Kaggle	Logistic Regression, 98.5% Accuracy
(Kaur, Kumar and Kumaraguru, 2019)	12 machine learning algorithms	Kaggle, Reuters, News Trends	Logistic Regression, Linear SVC

the best performing. Comparative analysis of supervised learning for fake news detection is provided also in tabular form below:

Table 2. 1 Comparative Analysis of Supervised Learning Methods

From this chapter, we developed an understanding of existing research works in the field of AI for the fake news detection problem. However, few studies were focussed on the context of the news, and image published within the news articles. Most of the research work used text data of news and modelled machine learning algorithms or deep learning methods on that for the task of fake news detection. We could observe GloVe, Word2Vec, and TF-IDF are NLP techniques implemented on the text data, and Decision Tree, Random Forest, XGBoost, Naïve Bayes, Logistic Regression, Support Vector Machine, and LinearSVC are machine learning methods mostly used for fake news detection. In case of deep learning methods, ANN, CNN, RNN, GRU, LSTM, Bi-Directional LSTM are mostly used. A few comparative studies on fake news detection methods have been conducted as well, but most of them used a few selective machine learning algorithms, not taking any deep learning methods into consideration. In our study, we will conduct a comparative analysis of fake news detection methods using all thirteen above mentioned machine learning and deep learning methods.

3 Research Approach

In this chapter, we will discuss the machine learning, deep learning and natural language processing methods that form the basis of this study. First chapter explains the concepts of machine learning algorithms suitable for fake news detection task, the second chapter covers the deep learning methods. In the third chapter, we will understand natural language processing methods, and lastly, we will discuss training the machine learning model and performance metrics.

3.1 Machine Learning

Machine learning is the field of study that gives computers the ability to automatically learn and improve from provided data without being comprehensively programmed. In machine learning, data is fed into a system, and chosen algorithms based on training on that dataset predicts the best possible outcomes.

Machine Learning problems can be divided into two distinct categories: Unsupervised Learning and Supervised Learning. In unsupervised learning problems, we are given only input data, and the aim is to find regularities in input, whereas in supervised learning problems, we are given a data set and the machine learning task is to learn the mapping from the input to the output (Alpaydın, 2014). Approach in supervised learning problems can be stated as follows:

$$\boldsymbol{Y} = \boldsymbol{g}(\boldsymbol{X}\,|\boldsymbol{\theta})$$

Where $\boldsymbol{X}$ is input, $\boldsymbol{g(.)}$ is model, $\boldsymbol{\theta}$ are parameters, and $\boldsymbol{Y}$ is output.

Machine Learning algorithms for fake news detection identified in chapter 2 of Related Work are all supervised learning methods. Next, we will discuss selected algorithms namely Decision Tree, Random Forest, XGBoost, Naïve Bayes, Logistic Regression, Support Vector Machine and LinearSVC in this section ahead.

3.1.1 Decision Tree

Decision Tree is a supervised learning method used for classification and regression tasks of machine learning. Decision Tree is a predictor algorithm, $\boldsymbol{h} : \boldsymbol{X} \rightarrow \boldsymbol{Y}$ predicts the label associated with input values x by travelling from a root node to a leaf of a tree. At each node, the successor child is chosen based on one of the features of $\boldsymbol{x}$ or

on a predefined set of splitting rules (Shalev and Ben, 2014). One of the drawbacks of decision tree algorithm is it could easily overfit.

3.1.2 Random Forest

Random Forest is a classifier introduced by Brieman (2001). It constructs an ensemble of decision trees to reduce the problem of overfitting.

In Random Forest, each tree is constructed by applying algorithm on training set S with an additional random vector, θ, where θ is sampled independent and identical distribution. Prediction output is assessed by majority predictions of individual trees (Shalev and Ben, 2014).

3.1.3 XGBoost

XGBoost or eXtreme Gradient Boosting is a decision tree-based machine learning supervised algorithm. It uses a gradient boosting framework and has inbuilt cross-validation method at each step, and in addition, it uses regularization to prevent overfitting.

This makes XGBoost one of the fastest and accurate algorithms. It can be applied in both cases of supervised learning problem i.e., regression, and classification.

3.1.4 Logistic Regression

Logistic Regression is a classification algorithm. It is a method to predict output as a binary variable, Y, where $Y \in \{0,1\}$ when we have one or more independent input variables, X. It is based on sigmoid function or logistic function

$$\boldsymbol{y = \frac{1}{1 + e^{-x}}}$$

If we consider, conditional mean of Y as $\pi(x)$ given X = x, and β are parameters then logistic regression (Larose, 2006) can be stated as:

$$\boldsymbol{\pi(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}}$$

Thus, $\pi(x)$ can be interpreted as probability with $\boldsymbol{0 \leq \pi(x) \leq 1}$

3.1.5 Naïve Bayes Estimation

Naïve Bayes algorithm is a supervised learning algorithm and used for classification tasks. It assumes that every feature in input data, *X*, is independent of each other, then the conditional probability $\boldsymbol{P(Y|X)}$ of output class *Y,* (Larose, 2006) can be stated as:

$$\boldsymbol{P(Y|X) = \frac{P(Y|X)P(Y)}{P(X)}}$$

Naïve Bayes algorithm works well with large datasets and can bring output quickly. It is, therefore, used in real-time prediction problems like spam-filtering.

3.1.6 Support Vector Machine (SVM)

SVM machine learning algorithm can be used for both classification and regression. SVM method searches for 'large margin' separator or hyperplane between the training data. Hyperplane is generated in a way so that training data is on the correct side and as far as possible from it (Shalev and Ben, 2014).

If x_i is input feature with set of weight w (or $\boldsymbol{w_i}$) for output value $\boldsymbol{y_i}$ with $\boldsymbol{b}$ as bias term then hyperplane can be defined by

$$\boldsymbol{w * x_i + b \geq +1, when\ y_i = +1}$$

$$\boldsymbol{w * x_i + b \geq -1, when\ y_i = -1}$$

In two-dimensional space, SVM can be a great classifier, and can be used for anomaly detection as well.

3.1.7 Linear SVC

Linear Support Vector classifier (SVC) is similar to the SVM. It generates a hyperplane between the training data like SVM that categorizes data. Linear SVC uses a linear kernel unlike SVM where Gaussian kernel is used. This makes Linear SVC less noisy in terms of bias and variance, but it can be unbalanced as hyperplane generate is always a straight line. Depending on the task at hand, it can be used for both regression and classification problems.

3.2 Deep Learning

Deep learning is a subset of artificial intelligence where with multiple layers of neural networks machine learn from data using a general-purpose learning procedure. (LeCun, Bengio and Hinton, 2015). Deep learning helps to bring out intricate patterns and being implemented in studies of biomedical research, image recognition, particle accelerator, and natural language processing.

In this section, we will discuss selected deep learning methods studied in literature review for the task of fake news detection.

3.2.1 Artificial Neural Network

In Artificial Neural Network (ANN), neural network consists of one input layer, one output layer, and one or more hidden layers in between the input and output layer. Also, hidden layers can have more neurons than input layer. Functioning of neural network uses feedforward neural network architecture and can be described in the following steps:

Step1: Input layer can have one or more input variables $\boldsymbol{x_i, x_2 \ldots . x_n}$ and based on respective weights $\boldsymbol{w_i, w_2 \ldots . w_n}$ data gets passed onto the hidden layer through synapse. Weights can be adjusted through backpropagation, and Stochastic Gradient Descent (SGD).

Step 2: In the neuron of the next layer, all the values get added up.

$$\sum_{i=1}^{n} \boldsymbol{w_i x_i}$$

Step3: In this step, an activation function, $\boldsymbol{\phi}$, is applied on aggregated weighted sum.

$$\boldsymbol{\phi}(\sum_{i=1}^{n} \boldsymbol{w_i x_i})$$

And based on the type of activation function, neurons get to decide to pass or not pass signal to the next hidden or output layer.

Activation Function: There are four types of activation function relevant to neural network illustrated below:

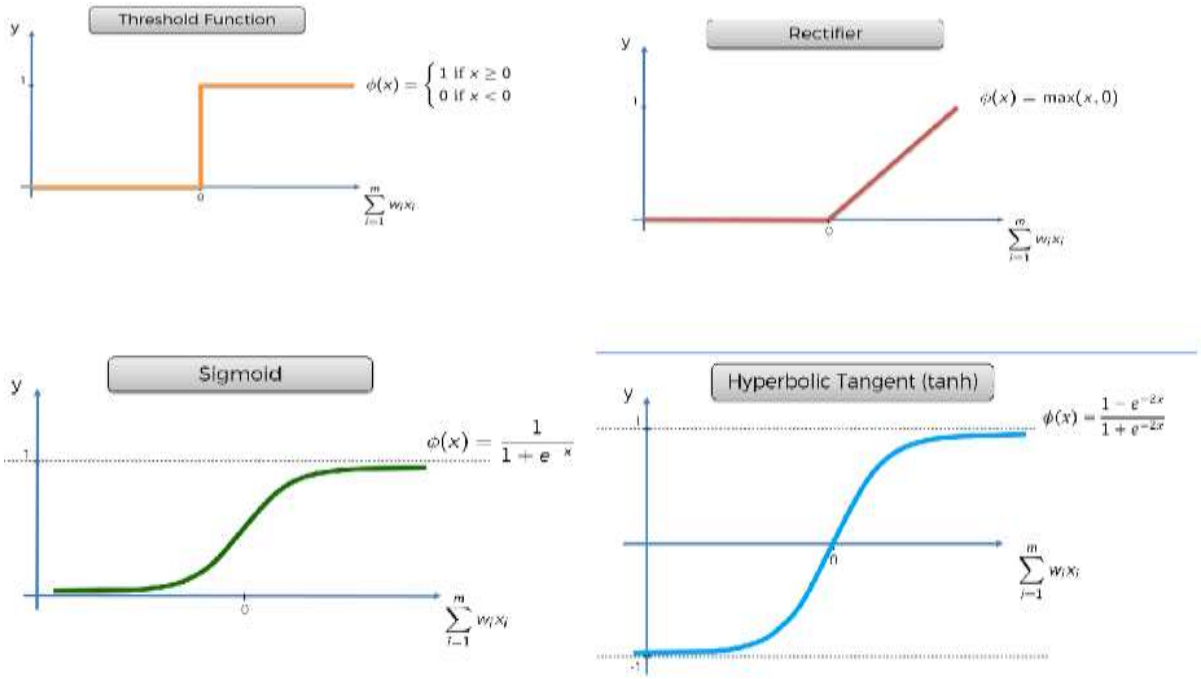


Figure 3. 1 Activation Function Source: Udemy

At present, the most popular activation function is rectified linear unit (ReLU) which is simply the half-wave rectifier $\boldsymbol{f(z) = max(z, 0)}$ (LeCun, Bengio and Hinton, 2015), For binary classification, Sigmoid activation function is implemented.

Step 4: In this last step, final information is passed from the last hidden layer to output layer. In output layer, output variables, $\boldsymbol{y_1, y_2 \ldots . y_n}$, can be continuous, binary, or categorical.

Step 5: In this final step, based on calculation, loss function is propagated back into the network to update the weights.

At this stage, we need to discuss Loss Function, Backpropagation, and Stochastic Gradient Descent before proceeding further.

Loss Function: Loss function is a function that measures the difference between actual target value and predicted outcome. If $\hat{\boldsymbol{y}}$ is the output value, and $\boldsymbol{y}$ is the target value, then loss function in case of binary cross entropy loss can be represented as:

$$\boldsymbol{L = -(\,p(x) * log\, q(x) + \big(1 - p(x)\big) * log(1 - q(x))}$$

where $p(x)$ is probability of class x in target, $q(x)$ is probability of class x in prediction.

Performance of the model is inversely proportional to the loss function. Lower the loss function, better the performance of the model. Another popular loss function is mean squared error loss for regression purposes.

Backpropagation: In backpropagation, information is fed back into the network from right to left. Based on the loss function generated, the set of weights has to be updated. We also must specify a 'learning rule' to update weights so that actual output would approximate desired output (Werbos, 1990).

Stochastic Gradient Descent: Firstly, we will discuss gradient descent before proceeding to stochastic gradient descent. Gradient descent is an iterative optimisation method to solve the problem of minimising loss function. In each iteration, we descend in the direction of negative gradient, with learning rate $\boldsymbol{\alpha}$ from $\boldsymbol{w^0}$ with the aim of converging to local minimum and update $\boldsymbol{w^0}$ **t**o $\boldsymbol{w^1}$ at each step (Watt, Borhani and Katsaggelos, 2016) Gradient descent can be shown as:

$$\boldsymbol{w^k = w^{k-1} - \alpha_k \nabla_g(w^{k-1})}$$

In above equation, $\boldsymbol{w^k}$ is weight parameter for *k*th gradient descent, $\boldsymbol{\alpha_k}$ is learning rate**.** One of the drawbacks of gradient descent is it requires cost function to be a convex function. Below given illustration is to demonstrate that:

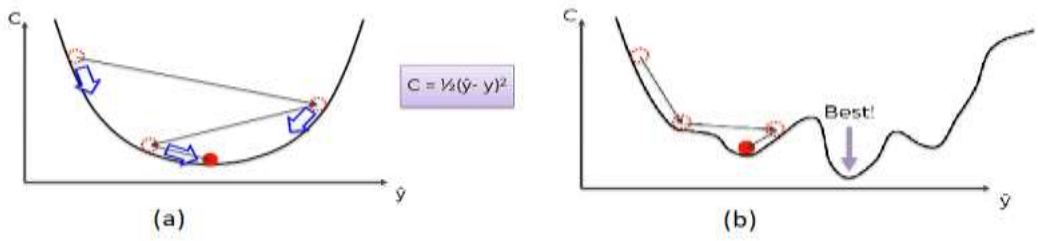


Figure 3. 2 Gradient Descent Source: Udemy

Figure (a) illustrates gradient descent converging at global minimum, Figure (b) illustrates gradient descent failing to converge when loss function is not convex.

Stochastic gradient descent does not require loss function to be convex because unlike gradient descent it does not use complete set of input row to update weights, but the stochastic process $\{w_t, t = 1, \ldots.\}$ instead uses randomly picked single

example z_t at each iteration (Bottou, 2010). Stochastic gradient descent is faster than gradient descent too.

3.2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is a second type of neural network, and widely used in the field of computer vision and machine learning problems.

In CNN, the first layer is the convolution layer which takes the input image. Then information gets passed through a series of layers. After convolution layer information gets passed to the pooling layer, then normalization layer, a fully connected layer, and loss layer. It can be illustrated as below:

$$\boldsymbol{x^1} \rightarrow [\boldsymbol{w^1}] \rightarrow \boldsymbol{x^2} \rightarrow \cdots \boldsymbol{x^{L-1}} \rightarrow [\boldsymbol{w^{L-1}}] \rightarrow \boldsymbol{x^L} \rightarrow [\boldsymbol{w^L}] \rightarrow \boldsymbol{z}$$

In this equation, $\boldsymbol{x^1}$ and $[\boldsymbol{w^1}]$ are input and parameter at first layer which provides input $\boldsymbol{x^2}$ for the second layer. Going through layers, finally at fully connected layer, we achieve $\boldsymbol{x^L} \boldsymbol{\epsilon} \mathbb{R}^c$ which determines the final probabilities of $\boldsymbol{x^1}$ belonging to the $\boldsymbol{C}$ categories (Wu, 2017). Last layer, $\boldsymbol{z}$ is loss layer, that determines loss function as:

$$\boldsymbol{z} = \frac{\boldsymbol{1}}{\boldsymbol{2}} \|\boldsymbol{t} - \boldsymbol{x^L}\|^2$$

where $\boldsymbol{t}$ is the target value and $\boldsymbol{x^L}$ is the predicted output.

Functioning of CNN can be described in following steps:

Step 1: Input image is processed with feature detector or kernel to generate feature maps. Many feature maps are generated at the first layer of convolution.

Step 2: In this step, to increase non-linearity we apply activation function (Kuo, 2016), usually ReLU.

Step 3: To achieve spatial invariance by reducing the resolution of feature maps (Scherer, Müller and Behnke, 2010), pooling operation is performed at this layer.

Step 4: In this step, we flatten pooled feature maps into a column. This column is then passed into ANN.

Step 5: In the fully connected layer, output from step 4 is provided as an input layer. Hidden layers in this neural network are fully connected and this predicts desired categorical output.

CNN is in a way convolutional layer, and pooling layer is addition to artificial neural network, and highly used in detection, segmentation, and image recognition.

3.2.3 Recurrent Neural Network

Recurrent Neural Network (RNN) is another type of neural network. It is a supervised learning algorithm, and widely used in the time series analysis and natural language processing problems.

Architecture of RNN is different from neural network. In RNN, hidden layers are grouped together, and it not only gives output but also gets input from neuron at previous time step. Distinction between ANN and RNN is illustrated below:

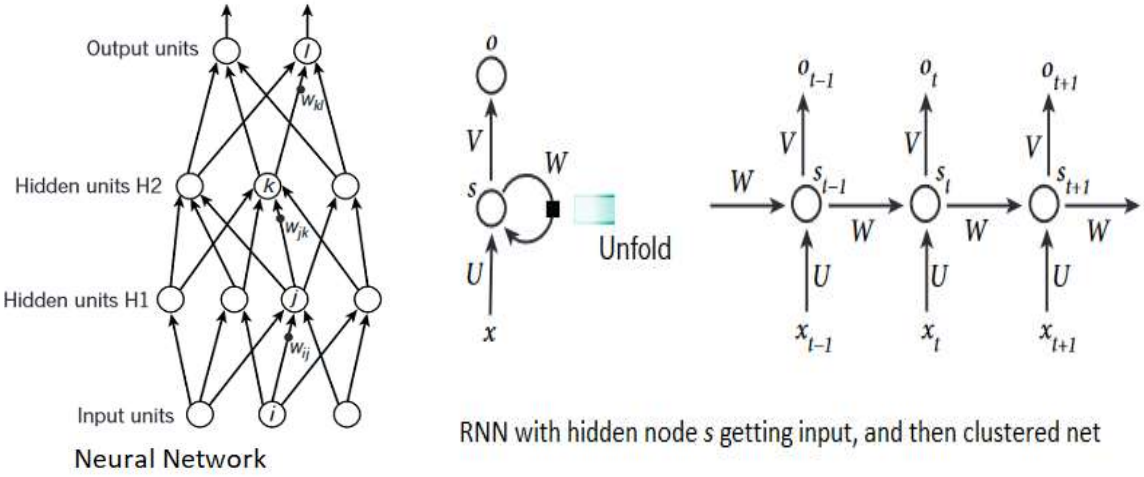


Figure 3. 3: Recurrent Neural Network source: (LeCun, Bengio and Hinton, 2015)

From image, we can understand that, both H1, and H2 hidden unit of neural network is grouped and hidden under node $\boldsymbol{s}$ in RNN with value $\boldsymbol{s_t}$ at time $\boldsymbol{t}$ (LeCun, Bengio and Hinton, 2015). Each node in the recurrent neural network gets output from other neurons at the previous time step. In this way, it can be stated that any output $\boldsymbol{o_{t+1}}$ at any time step $\boldsymbol{t+1}$ is influenced by all input values $\boldsymbol{x_t, x_{t-1}..x_{t-n}}$ from previous time steps. Backpropagation algorithm could be implemented from right to left back into RNN to improve model performance.

Architecture of RNN can be of varied type based as illustrated below:

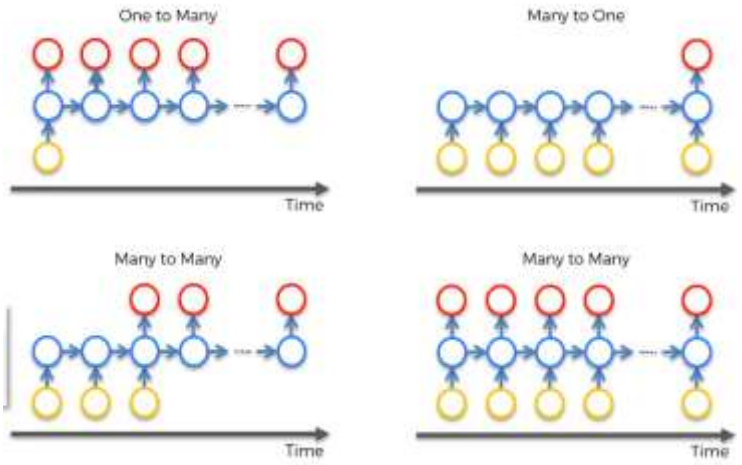


Figure 3. 4: Different architecture of RNN Source: Udemy

Based on machine learning problems at hand, different types of RNN architecture can be deployed. For instance, in case of music generation problem, one-to-many architecture and for sentiment classification many-to-one architecture can be implemented.

RNN is a very efficient neural network system and meant to learn long-term dependencies. It can be represented as:

$$\boldsymbol{x_t = W_{rec}\sigma(x_{t-1}) + W_{in}u_t + b}$$

In the equation, $\boldsymbol{W_{rec}}$ is recurrent weight**,** $\boldsymbol{\sigma}$ is element-wise function, $\boldsymbol{W_{in}}$ is input weight, $\boldsymbol{u_t}$ is input, $\boldsymbol{b}$ is bias, and $\boldsymbol{x_t}$ is state at time $\boldsymbol{t}$**.** (Pascanu, Mikolov, and Bengio, 2013)

However, training deep RNN could be difficult because all the nodes share the same weight (LeCun, Bengio and Hinton, 2015). This way when weight, $\boldsymbol{W_{rec}}$, is small, backpropagating gradient could start to vanish over time, making nodes at earlier time step, $\boldsymbol{s_{t-n}}$, impossible to learn. This phenomenon is called Vanishing Gradient problem. There is another problem called Exploding Gradient when weights are large and backpropagating gradients start to grow exponentially. Exploding gradient can be solved by gradient clipping. For vanishing gradient, we will need to understand LSTM and GRU.

3.2.4 Long Short-Term Memory

Long Short-Term Memory (LSTM) is developed to solve vanishing gradient problem.

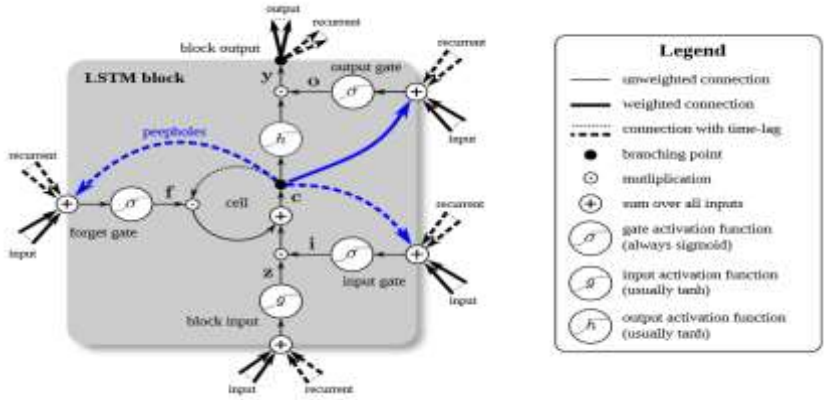


Figure 3. 5: LSTM source: (Greff et al., 2017)

In LSTM architecture, a concept of a memory cell has been introduced to act like an accumulator. After that, it has three inputs (first input is input at that time step, second from previous neuron, and third one from memory cell), and two outputs (one as output, one to be fed back into the system). Also, it has three gates – input gate, forget gate and output gate. Sigmoid function $\boldsymbol{(\sigma)}$ is used as gates activation function, whereas hyperbolic tangent ($\boldsymbol{tanh}$)is used as input and output activation function (Greff et al., 2017). LSTM can be understood better with equation as below:

$$\boldsymbol{c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t}$$

$$\boldsymbol{\tilde{c}_t = g(W_c x_t + U_c h_{t-1} + b_c)}$$

$$\boldsymbol{h_t = o_t \odot g(c_t)}$$

Where $\boldsymbol{c_t}$ is current memory cell, $\boldsymbol{f_t}$ is forget gate, $\boldsymbol{c_{t-1}}$ is memory cell at previous time step, $\boldsymbol{i_t}$ is input gate, $\boldsymbol{\tilde{c}_t}$ is an upcoming memory cell. Also, $\boldsymbol{g}$ is element-wise function, $\boldsymbol{W}$, $\boldsymbol{U}$, and $\boldsymbol{b}$ are weights and bias. $\boldsymbol{h_t}$ is hidden state at time $\boldsymbol{t}$ and $\boldsymbol{o_t}$ is the output gate. Important, $\odot$ is element-wise multiplication for weighted sum (Dey and Salem, 2017). With same notations, gates can be explained as:

$$\boldsymbol{i_t = \sigma (W_i x_t + U_i h_{t-1} + b_i)}$$

$$\boldsymbol{f_t = \sigma (W_f x_t + U_f h_{t-1} + b_f)}$$

$$\boldsymbol{o_t = \sigma (W_o x_t + U_o h_{t-1} + b_o)}$$

LSTM architecture with RNN is proven to store long-term dependencies and thus to solve the vanishing gradient problem.

3.2.5 Gated Recurrent Units

Gated Recurrent Units (GRU) is another architecture to solve the vanishing gradient problem of RNN.

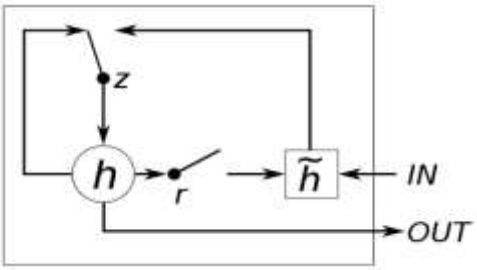


Figure 3. 6: Source: (Cho, Chung, Gulcehre and Bengio, 2014).
r and *z* are reset & update gates

In GRU architecture, there is no memory cell, and it has only two gates called update gate and reset gate. Update gate decides how much the unit updates its information or adds new. Reset gate decides how much the unit forgets previously computed information (Cho, Chung, Gulcehre and Bengio, 2014). GRU can be represented with these equations:

$$\boldsymbol{h_t} = (\mathbf{1} - \boldsymbol{z_t}) \odot \boldsymbol{h_{t-1}} + \boldsymbol{z_t} \odot \widetilde{\boldsymbol{h_t}}$$

$$\widetilde{\boldsymbol{h_t}} = \boldsymbol{g}(\boldsymbol{W_h x_t} + \boldsymbol{U_h}(\boldsymbol{r_t} \odot \boldsymbol{h_{t-1}}) + \boldsymbol{b_h})$$

$$\boldsymbol{z_t} = \boldsymbol{\sigma}\,(\boldsymbol{W_z x_t} + \boldsymbol{U_z h_{t-1}} + \boldsymbol{b_z})$$

$$\boldsymbol{r_t} = \boldsymbol{\sigma}\,(\boldsymbol{W_r x_t} + \boldsymbol{U_r h_{t-1}} + \boldsymbol{b_r})$$

Notations are similar to LSTM architecture. In addition, we have here $\boldsymbol{r}$ as reset gate and $\boldsymbol{z}$ as update gate. GRU is faster to train than LSTM, however, both are equally reliable in solving the vanishing gradient problem to get better predictions.

3.2.6 Bi-Directional Long Short-Term Memory

Bi-Directional LSTM architecture is like LSTM, but instead of one LSTM it trains two. Second one is in reverse direction towards input. To understand Bi-Directional LSTM, we need to understand B-Directional Recurrent Neural Network (BRNN) first. BRNN is developed with the idea to capture future input information coming at later than current time frame $\boldsymbol{t_f}$ in RNN. In RNN, predictions can be delayed for certain time $\boldsymbol{t_{f+n}}$, but leaves less modelling power to combine predictions and inputs.

BRNN architecture has two split neuron states. One is for positive time direction called forward state, another is for negative time direction called backward state. Importantly, output from forward state is not fed into backward state, and similarly no output is fed into forward state from backward state (Schuster and Paliwal, 1997).

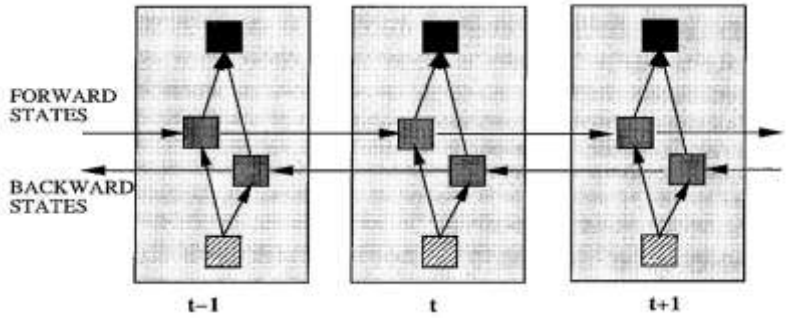


Figure 3. 7: BRNN Source: (Schuster and Paliwal, 1997).

Thus, Bi-Directional LSTM based on BRNN can be very effective in natural language processing problems like sentiment analysis, sentence classification where complete sentences are provided. Only disadvantage with BRNN is that it needs the entire sequence of inputs to be available before starting. So, BRNN and thus, Bi-Directional LSTM cannot be utilized in speech-recognition problems.

3.3 Natural Language Processing

Natural Language Processing (NLP) is subset of artificial intelligence. In NLP, we deal with text data and objective here is to make computers read, understand, and generate human-like natural language. NLP techniques are implemented in problems of sentence classification, sentiment analysis, trend analysis, and review analysis. In this section, we will understand the basic concept of text data pre-processing and text feature extraction in natural language processing required for this study.

3.3.1 Text Data Pre-processing

Tokenization: Tokenization is a process of breaking up the original sequence of text into component pieces called tokens, throwing away certain characters such as punctuation. (Tokenization, 2020). Tokenization can be illustrated as below:

Input	Come on! Finish this, we have to code as well!									
Output	Come	On	Finish	This	We	Have	To	Code	As	Well

Table 3. 1 Tokenization

Stop-Words: Stop words are those insignificant words that are used to connect meaning or sentence structure. For example, articles, preposition, conjunctions such as a, an, the, to, at, by, but, yet, this, that, until, what, where, how, etc can be called as stop-words. These stop words would create noise in text feature extraction, and thus must be removed before the next step.

3.3.2 Text Feature Extraction

In NLP, the text-feature extraction technique computes the weight of words in each text sequence. This technique focuses on the occurrences and weight of each word, ignoring sentence structure, disregarding grammar, or word order.

Count Vectorization: Count Vectorization is another text-feature extracting technique. In this technique, we count the occurrences of words and present it in a Document-Term Matrix (DTM).

For instance, ["David likes to play football.", "Mary likes football too"] will be represented as:

David	Likes	To	Play	Football	Mary	Too
1	1	1	1	1	0	0
0	1	1	0	1	1	0

Table 3. 2 DTM Count Vectorization

Term Frequency-Inverse Document Frequency (TF-IDF): TF-IDF is a process of text feature extraction to DTM. In TF-IDF, frequency of words in a corpus is rescaled so that frequently appearing words could be penalized, and infrequent words could be given appropriate weightage.

Term Frequency (TF) is just raw count of a term in a document and can be stated as:

$$tf(t,d) = \frac{the\ number\ of\ occurence\ of\ term\ t\ in\ document\ d}{the\ number\ of\ all\ words\ in\ document\ d}$$

Inverse Document Frequency (IDF) is a log inverse fraction of the documents that contain the word, and can be stated as:

$$idf(t,d) = \log \frac{the\ total\ number\ of\ documents\ in\ retrieved\ set}{the\ number\ of\ documents\ indexed\ by\ term\ t + 1}$$

TF-IDF can be represented as: $tfidf = tf(t,d) * idf(t,d)$

TF-IDF will be slightly high if a term is high frequency for a given document and low frequency for the whole set of retrieved documents (Qiu, Jiang and Chen, 2020)

3.4 Training the Model

Training of machine learning models starts with dataset collection. After that, next step is data cleaning and pre-processing. Redundant features have to be removed in this step. Third step is generalization. It trains a machine learning algorithm to learn from existing data so that it can map the unseen inputs as well.

Generalisation: In generalisation, after splitting data into two or three sets of training data, test data, and validation data, we select a machine learning algorithm to learn from training data and try to make accurate predictions on the test set. In certain cases, prediction on a test set fails to be accurate, because of two reasons, overfitting, and underfitting. Overfitting happens when the model is trained too well on the training set, whereas underfitting happens when the model is not sufficiently trained on the training set. Problem of overfitting can be avoided by using some of the following: removing some features, training on more data, using cross-validation set, and adding or increasing regularization term. Similarly, problem of underfitting can be avoided by decreasing the value of regularization term and increasing the number of features.

Performance Evaluation: Machine learning model performance can be summarised by a technique called confusion matrix. Confusion matrix is a matrix of row and column where actual values, and predicted values are represented. We can precisely deduce the kind of error a classification model is making through a confusion matrix. In Figure 3.8 of confusion matrix on next page, TP stands for 'True Positive'. It means predicted value is positive and it is correctly predicted. FP stands for 'False Positive'. It means the predicted value is negative and it is correctly predicted. FP stands for 'False Positive.' It means predicted value is positive, but it is incorrectly predicted. TN stands for 'False Negative'. It means the predicted value is negative, but again it is incorrectly predicted.

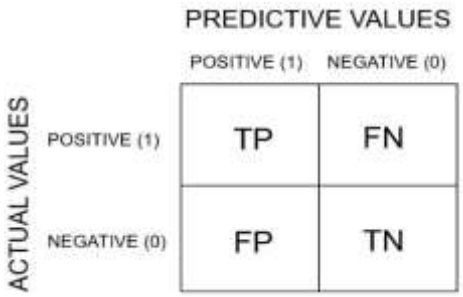


Figure 3. 8 Confusion Matrix Source: Google

Performance Metrics: Performance of a machine learning classifier can be adjudged through four metrics of Accuracy, Precision, Recall, and F1 score explained below:

Accuracy: It is a ratio of total correctly predicted values to the total number of values.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision: It is a ratio of total number of correctly predicted positive values to the total number of predicted positive values.

$$Precision = \frac{TP}{TP + FP}$$

Recall: It is a ratio of total number of correctly predicted positive values to the total number of actual numbers of positive values.

$$Recall = \frac{TP}{TP + FN}$$

F1 Score: F1 score is a harmonic mean of Precision and Recall.

$$F1\ Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

In this chapter, we discussed the algorithms of machine learning relevant to this study in the first section. Then in the second section, we discussed concepts of deep learning that would be implemented for fake news detection. In the third section, we learnt natural language processing methods required. Lastly, we discussed training machine learning models and evaluating classification reports on different performance metrics.

4 Research and Results

In this chapter, we will implement our understanding of machine learning, deep learning, and natural language processing concepts to conduct our research. In the first section, we detailed the methodology of our work, and technologies to be used in this study. Second section is to provide insights into the dataset collected. In the third section, details of data pre-processing, and feature extraction is provided. Forth and fifth section provided details of machine learning and deep learning modelling and collected results. Lastly, in the sixth section, we compare our fake news detection models on their performances.

4.1 Methodology

This section describes our methodology for developing our models for comparative analysis. First, we are going to collect a labelled dataset. Then data pre-processing will be applied to remove redundant features from the dataset and/or labelling certain features. Next, text data pre-processing techniques like stop words removal, and punctuation removal, will be implemented. After that, selected machine learning algorithms namely: Decision-Tree, Random-Forest, Logistic Regression, Naïve Bayes, Support Vector Machines (SVM), and XGBoost will be modelled. We will also be testing our dataset with deep learning models like Artificial Neural Network (ANN), Convolution Neural Network (CNN), Simple Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Reccurent Units (GRU), and Bi-directional LSTM. Lastly, we will compare our results based on performance metrics, and evaluate performance of our models. Process of our work can be illustrated as below:

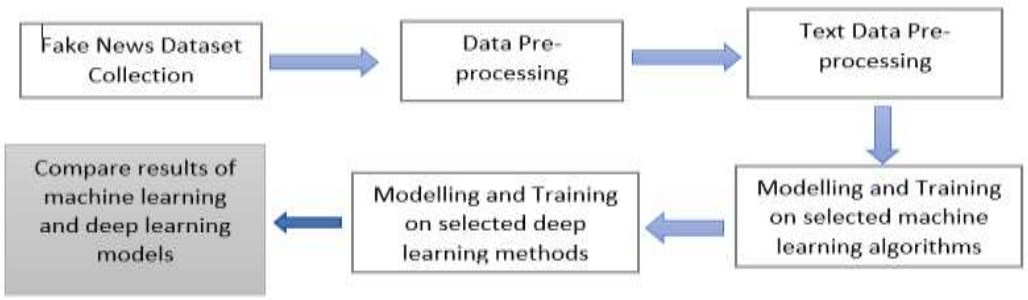


Figure 4. 1 Flowchart of Our Research Work

4.1.1 Technologies Used

In our study of existing literature for fake news detection models and comparative analysis in chapter 2, we came to understand which technologies would be the most suitable to conduct our own study. These are Python, Pandas, Scikit-Learn, Natural Language Toolkit (NLTK), and Tensorflow.

Python: Python is a programming language with simple syntaxes. It has availability of brilliant libraries to conduct any artificial intelligence research work. Python will be used to write code and import libraries such as Pandas, Scikit-Learn, natural language toolkit (NLTK), Collection, Tensorflow and Seaborn to be used in this study.

Pandas: Pandas is a data analysis library of Python. It allows reading, writing, merging, joining, replacing, and reshaping datasets proficiently. Pandas is used in this study to clear the dataset with redundant features, and data pre-processing.

Scikit-Learn: Scikit-learn is a machine learning library of Python. It is simple, efficient, and can be used for various machine learning problem ranging from classification, clustering, and regression. It also has feature extraction preprocessing like TF-IDF. Scikit-Learn will be used for modelling machine learning algorithms for fake news detection as explained in chapter 3.

Natural Language Toolkit (NLTK): NLTK is a package for Natural Language Processing (NLP) required for our study. We will be using NLTK to remove stop words, and punctuations from news text data.

Tensorflow: Tensorflow is a deep learning library focussed on training and inferencing of deep neural networks. Multiple hidden layers with max pooling, backpropagation, stochastic gradient descent, and loss function are essential to model neural network such as CNN, RNN, LSTM, GRU and Bi-Directional LSTM as explained in chapter 3, and can be easily created with Tensorflow.

Seaborn: Seaborn is a data visualization library based on matplotlib. It helps to generate most informative statistical graphics. We will be using Seaborn to create visualization while evaluating our fake news detection models.

4.2 Data Collection

For this study, we need a publicly available labelled dataset. In our study of existing research on fake news detection in chapter 2, many different datasets were used. Details of datasets are given below in Table 4.1.

Dataset	Elements		Citation
	Quantity	Category	
Liar, Liar, Pants-on-Fire	1,050	Pants-onFire	(Wang, 2017)
	2,063-2,638	Other Five	
Buzzfeed	4,111	Real News	(Reis et al., 2019)
	110	Fake	
	308	Satire	
UNBiased	1,400	Fake	(Gravanis, Vakali, Diamantaras, and Karadais, 2019)
	2,004	Real News	
News Trend	3,171	Real News	(Kaur, Kumar, and Kumaraguru, 2019)
	3,164	Fake	
Kaggle	1,865	Real News	(Poddar, Amali D, and Umadevi, 2019)
	2,118	Fake	
Reuters	9,622	Real News	(Kaur, Kumar, and Kumaraguru, 2019)
	10,347	Fake	

Table 4. 1 Datasets used for Fake News Detection Models

Among all the dataset detailed above, we decided on "Liar, Liar, Pants on Fire" (LIAR) dataset as we needed a dataset with more than 10,000 columns. It will help to prevent the undersampling problem and obtain reliable results. This dataset has 12.8K columns of news statement, and proportionally divided into three section of training set, validation set, and test set, each having 13 columns (Wang, 2017). Classification of news in this dataset has been done into six Labels: True, Mostly True, Half True, Barely True, False, and Pants Fire. It can be seen in below given figure:

	File Type	Label	Statement	Context	Speaker	Position	State	Party	n1	n2	n3	n4	n5	Source
0	2635.json	false	Says the Annies List political group supports ...	abortion	dwayne-bohac	State representative	Texas	republican	0.0	1.0	0.0	0.0	0.0	a mailer
1	10540.json	half-true	When did the decline of coal start? It started...	energy,history,job-accomplishments	scott-surovell	State delegate	Virginia	democrat	0.0	0.0	1.0	1.0	0.0	a floor speech.
2	324.json	mostly-true	Hillary Clinton agrees with John McCain "by vo...	foreign-policy	barack-obama	President	Illinois	democrat	70.0	71.0	160.0	163.0	9.0	Denver
3	1123.json	false	Health care reform legislation is likely to ma...	health-care	blog-posting	NaN	NaN	none	7.0	19.0	3.0	5.0	44.0	a news release
4	9028.json	half-true	The economic turnaround started at the end of ...	economy,jobs	charlie-crist	NaN	Florida	democrat	15.0	9.0	20.0	19.0	2.0	an interview on CNN

Figure 4. 2 Dataset before pre-processing

4.3 Data Pre-processing

In this section, we will conduct data pre-processing on our collected LIAR dataset.

Step 1: In this first step, we will shorten columns from thirteen to two. As we are conducting classification of news on the basis of the text, we require only the 'Statement' column and news 'Label'. A comparative analysis of five machine learning models on this same dataset (Agarwal, Sultana, Malhotra and Sarkar, 2019) studied in chapter 2 implemented a similar strategy.

Step 2: In this step, we will shorten six news labels under 'Label' into two 'Real' and 'Fake'.

Research Work	Label 'True'	Label 'False'
(Agarwal, Sultana, Malhotra and Sarkar, 2019)	True, Mostly-True Half-True	Barely-True, False, Pants-Fire
Our Approach	True, Mostly-True	Barely-True, False, Pants-Fire

Table 4. 2 Categorization of News Label

Reasoning for Our Approach:

As Half-True news is mathematical equivalent to half-false news or in general neutral news. However, in the context of news, any news that is considered neutral is basically a true news. Therefore, we concluded that news with 'Label' as 'Half True' will effectively fail to contribute to classification purpose of news into 'Fake' or 'Real' required for this study and thus can be discarded.

Label Pants-Fire have 1,050 cases, whereas other labels range from 2,063 to 2,638 (Wang, 2017). Thus, our dataset, with new categorization of six news labels into just two, remain well-balanced.

For performance metrics, Recall, explained in section 3.4, that would mean, 'Of all the fake news that are fake, how many we are correctly predicting' the number of fake news will remain the same in our approach as in the other study (Agarwal, Sultana, Malhotra and Sarkar, 2019).

Step 3: As explained in section 3.3.2 of chapter 3, we need to perform feature extraction on the text data of the 'Statement' column before passing into our models. This process will convert input text data as numerical.

In case of deep learning models, we have discussed that neural network learn to predict output when input data will be passed through hidden layers, backpropagated with loss function and improved with stochastic gradient descent. This deep learning process over a number of epochs will generate output in float values closer to classification labels. Therefore, classification labels must be in numbers, any number. This way deep learning models would learn through the training set to predict close to the expected number provided for the classification label.

Therefore, in this step 3, for the sake of simplicity, we will rename our label 'Real' and 'Fake' to '1' and '0' respectively.

After completing above mentioned three steps of data pre-processing on all the three sets, training set, validation set, and test set, and our dataset can be seen like this:

	Label	Statement
0	0	Says the Annies List political group supports ...
1	1	When did the decline of coal start? It started...
2	1	Hillary Clinton agrees with John McCain "by vo...
3	0	Health care reform legislation is likely to ma...
4	1	The economic turnaround started at the end of ...
...	...	...
10235	1	There are a larger number of shark attacks in ...
10236	1	Democrats have now become the party of the [At...
10237	1	Says an alternative to Social Security that op...
10238	0	On lifting the U.S. Cuban embargo and allowing...
10239	0	The Department of Veterans Affairs has a manua...

Figure 4. 3 Data after pre-processing

4.4 Modelling Machine Learning Algorithms

After data cleaning, now we have our training set, test set, and validation set left with two columns, 'Statement' and 'Label'. 'Label' column is with integer data type for classification purpose.

Step1: In this first step of modelling, we will create our X_train, y_train from 'Statement' and 'Label' columns of training set respectively. Similarly, we will create X_test, and y_test. Lastly, we will create X_valid, and y_valid from the 'Statement' and 'Label' columns of the validation set.

Step 2: Next, we will be performing text feature extraction on the text of 'Statement' column. Here, text feature extraction technique TF-IDF will be implemented to create document-term matrix (DTM). This process will create a DTM with accurate weight for each word in the corpus.

Scikit-Learn library has a function named TfidfVectorizer which converts corpus of raw text into a DTM. TfidfVectorizer also removes accents, punctuations, and stop words (if specified) (Scikit-learn.org, 2020). It performs stemming to a certain extent too. We will now transform our training data, X_train with TfidfVectorizer. Then, we will model data on different machine learning algorithms.

Decision Tree

In our fake news detection problem modelling with Decision Tree algorithm, we will use the Pipeline feature of Scikit-Learn to train our classifier, and then we will fit our train data X_train and y_train before predicting our final outputs. On performance metrics, our Decision Tree model scored as below given table:

Accuracy Score: 0.5198			
	Precision	Recall	F1 Score
Fake (0)	0.47	0.58	0.52
Real (1)	0.58	0.47	0.52

Table 4. 3 Decision Tree

Random Forest

In Random Forest model, we will implement using Pipeline feature to fit, and train our model with training set before predicting our outputs on test set as below given:

Accuracy Score: 0.6029			
	Precision	Recall	F1 Score
Fake (0)	0.51	0.71	0.59
Real (1)	0.70	0.41	0.52

Table 4. 4: Random Forest

XGBoost

In, XGBoost modelling, we passed our validation data i.e. X_valid and y_valid while fitting our model in the pipeline. We fit our model in Pipeline feature to train on the training set. Then, our evaluation on the test set is as follows:

Accuracy Score: 0.5735			
	Precision	Recall	F1 Score
Fake (0)	0.51	0.68	0.58
Real (1)	0.67	0.45	0.54

Table 4. 5: XGBoost

Naïve Bayes

Similar procedure as used in the above-mentioned models have been implemented with Naïve Bayes algorithm, before getting our predictions as below given table:

Accuracy Score: 0.5695			
	Precision	Recall	F1 Score
Fake (0)	0.51	0.68	0.58
Real (1)	0.70	0.40	0.51

Table 4. 6: Naïve Bayes

Logistic Regression

We train our model with Logistic Regression following the same steps of previous modelling. Our predicted outputs with accuracy score is given below:

Accuracy Score: 0.5924			
	Precision	Recall	F1 Score
Fake (0)	0.53	0.73	0.62
Real (1)	0.69	0.48	0.57

Table 4. 7: Logistic Regression

Support Vector Machine (SVM)

SVM is adjudged as one of the best algorithms in fake news detection tasks. With SVM, we are following the similar way of fitting and training of our model. Result obtained in confusion matrix and accuracy score on our test set is given below:

Accuracy Score: 0.5834			
	Precision	Recall	F1 Score
Fake (0)	0.52	0.76	0.62
Real (1)	0.70	0.45	0.54

Table 4. 8: SVM

Linear Support Vector Classifier

LinearSVC is the last algorithm to be implemented for fake news detection problem. Here, we will again fit our transformed data from TfidfVecorizer in the pipeline method with LinearSVC and develop our model. Our results are as follows:

Accuracy Score: 0.5685			
	Precision	Recall	F1 Score
Fake (0)	0.51	0.64	0.57
Real (1)	0.64	0.51	0.57

Table 4. 9: Linear SVC

4.5 Modelling Deep Learning Methods

Step 1: After data pre-processing, we have our train set, test set and validation set trimmed to two columns, i.e., Label, and Statement. Label column represents 'Fake' news as '0' and 'Real' news as '1'. In this step, we will be removing punctuation, and stop words with NLTK tool. Then, we will need the total count of individual words available in the corpus to pass into Tokenizer in the next step. To get this count, we will be concatenating columns of train set, test set, and validation set into a new set with Pandas concatenation function, and with python library collections, we will get our desired number of individual words in the dataset.

Step 2: Here, we will create our training data, training labels, test data, test labels, valid data, and valid labels from the pre-processed data-frames.

Step3: In this step, we will implement Keras, a neural networks library from Tensorflow. We will import *Tokenizer*, and *pad_sequence* function from text and sequence pre-processing of Keras. In *Tokenizer*, we will need the total number of words ascertained in the Step 1 for *num_words*, and in the *pad_sequence*, we will need *maxlen* or maximum length of sequence. (Wang, 2017) mentioned the average length of statement in each row is 17.9 in the paper, so, we will pass 20 as *maxlen* here.

Step 4: Here, we will create our padded sequences for the training set, test set, and validation set as train_padded, test_padded, and valid_padded. The train_padded and train_labels (created in step 2) will be used while training the model as training data. We will also use valid_padded, and valid_labels (created in step 2) as a cross-validation set to prevent overfitting.

Artificial Neural Network (ANN)

In our modelling of ANN, we are using six hidden layers. Activation function for the hidden layer is selected as Rectified Linear Unit (ReLU). Next, we need to pick an optimizer and loss function. Optimizer is selected as 'adam' and since this modelling is for classification tasks, we are using 'binary cross entropy' as our loss function. Our architecture is summarised in the Figure 4.4 below:

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_25 (Embedding)     (None, 20, 32)            404256
_________________________________________________________________
dense_66 (Dense)             (None, 20, 32)            1056
_________________________________________________________________
dense_67 (Dense)             (None, 20, 32)            1056
_________________________________________________________________
dense_68 (Dense)             (None, 20, 32)            1056
_________________________________________________________________
dense_69 (Dense)             (None, 20, 32)            1056
_________________________________________________________________
dense_70 (Dense)             (None, 20, 32)            1056
_________________________________________________________________
dense_71 (Dense)             (None, 20, 32)            1056
_________________________________________________________________
flatten_9 (Flatten)          (None, 640)               0
_________________________________________________________________
dense_72 (Dense)             (None, 1)                 641
=================================================================
Total params: 411,233
Trainable params: 411,233
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. 4: ANN architecture

While training our model, we are using valid_padded and valid_labels as our validation data, and 32 batch_size. Epoch is decided as 40 after many reruns of the model. In the output layer, we are getting predictions in float value between 0.00 and 1.00 for our class label of 0 and 1. So, for the better understanding of predictions, we are classifying it into Boolean value of 'Real' if more than 0.5, and then converting Boolean into integer to get our desired output as either '0' or '1'. Confusion matrix of our model performance and accuracy is presented in Table 4.9 below:

Accuracy Score: 0.5588			
	Precision	Recall	F1 Score
Fake (0)	0.60	0.59	0.59
Real (1)	0.50	0.52	0.51

Table 4. 10: ANN Result

Convolutional Neural Network (CNN)

In our CNN model, we are adding a convolution layer (Conv1D), a pooling layer(globalAveragePooling1D) before adding two final layers. Optimizer and loss function are selected as in the ANN. Our CNN architecture is presented on next page:

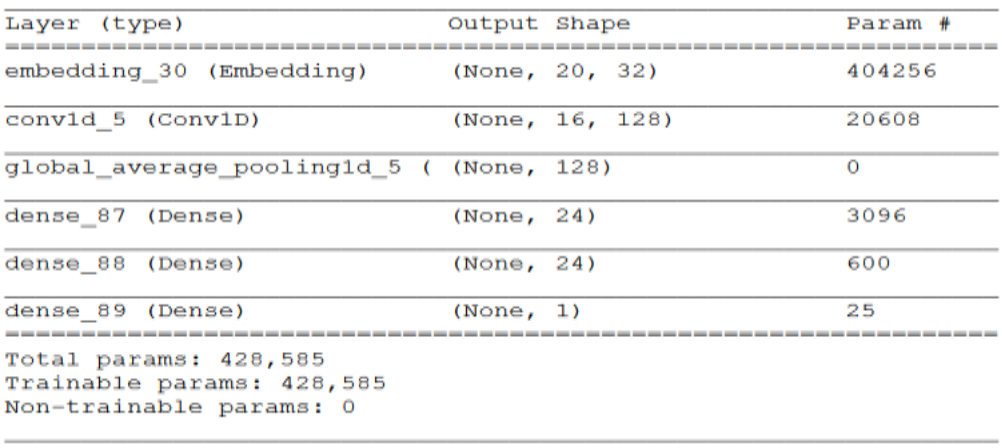


Figure 4. 5: CNN architecture

CNN model is trained as our ANN model with the same batch_size. However, we are getting better results with epochs 15 this time. Predictions are converted from float to integer as explained in ANN modelling, and our results on performance metrics are provided below in Table 4.10

Accuracy Score: 0.5658			
	Precision	Recall	F1 Score
Fake (0)	0.60	0.64	0.62
Real (1)	0.52	0.47	0.49

Table 4. 11: CNN Result

Recurrent Neural Network (RNN)

In our RNN model, we are using five input layers, and a flattening layer before the output layer. ReLU is our activation function for the hidden layers, and we are using 'adam' optimiser, and binary cross entropy loss function. Our model architecture is provided in the Figure 4.6 below.

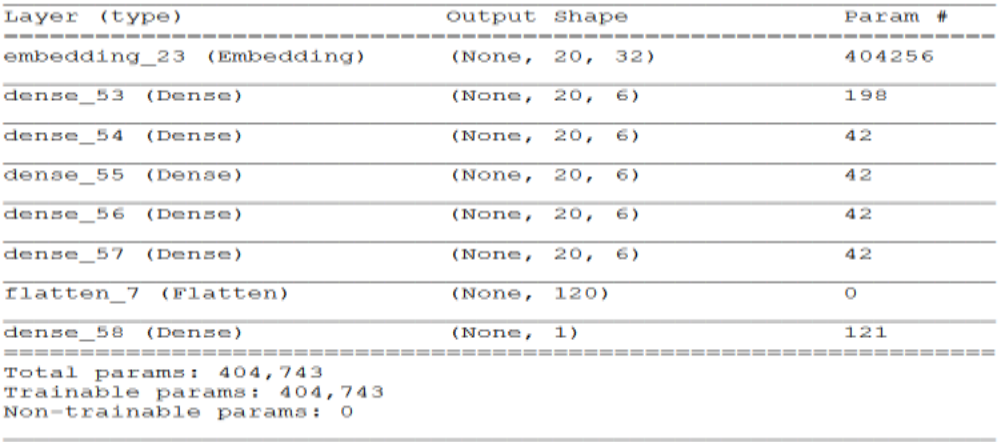


Figure 4. 6: RNN architecture

We found training 60 epochs and 32 batch size provides best results for RNN. Predictions are made similarly and illustrated in the Table 4.11

Accuracy Score: 0.5578			
	Precision	Recall	F1 Score
Fake (0)	0.61	0.57	0.59
Real (1)	0.51	0.54	0.52

Table 4. 12: RNN Result

Long Short-Term Memory (LSTM)

In our LSTM model, six hidden layers are used. We have selected 32 LTSM units and dropout as 0.1. The return_sequences has been set to 'True' for hidden layers. Activation function, optimizer, and loss function is again the same as the previous modelling. Our model architecture can be summarised in the Figure 4.7.

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 20, 32)            404256
_________________________________________________________________
lstm_20 (LSTM)               (None, 20, 32)            8320
_________________________________________________________________
lstm_21 (LSTM)               (None, 20, 32)            8320
_________________________________________________________________
lstm_22 (LSTM)               (None, 20, 32)            8320
_________________________________________________________________
lstm_23 (LSTM)               (None, 20, 32)            8320
_________________________________________________________________
lstm_24 (LSTM)               (None, 20, 32)            8320
_________________________________________________________________
lstm_25 (LSTM)               (None, 20, 32)            8320
_________________________________________________________________
lstm_26 (LSTM)               (None, 32)                8320
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 462,529
Trainable params: 462,529
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. 7:LSTM architecture

While modelling LSTM network, we found 25 epochs to be the best performing for predicting outputs. And predictions are again accumulated into '0' or '1' as previously discussed. Our LSTM network provided us below given result.

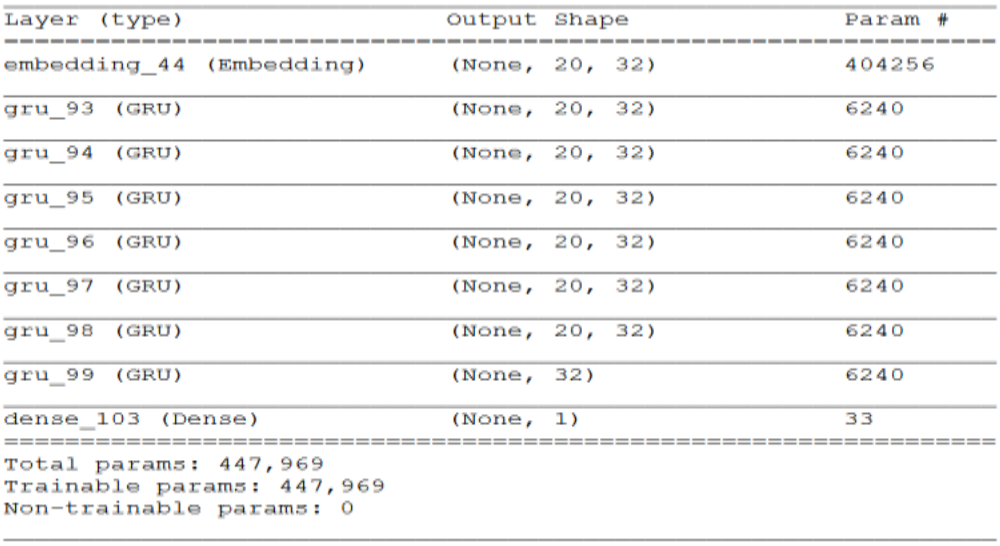
Accuracy Score: 0.5858			
	Precision	Recall	F1 Score
Fake (0)	0.62	0.66	0.64
Real (1)	0.54	0.50	0.52

Table 4. 13: LSTM Result

Gated Recurrent Units (GRU)

In our GRU network model for news classification, we are using 6 hidden GRU layers, but with 32 units in each layer. Activation function for hidden layers is 'ReLU' and for

the output layer is 'sigmoid'. Optimizer and loss function is like previous modelling. Architecture for the GRU model is provided below in the Figure 4.8.



Layer (type)	Output Shape	Param #
embedding_44 (Embedding)	(None, 20, 32)	404256
gru_93 (GRU)	(None, 20, 32)	6240
gru_94 (GRU)	(None, 20, 32)	6240
gru_95 (GRU)	(None, 20, 32)	6240
gru_96 (GRU)	(None, 20, 32)	6240
gru_97 (GRU)	(None, 20, 32)	6240
gru_98 (GRU)	(None, 20, 32)	6240
gru_99 (GRU)	(None, 32)	6240
dense_103 (Dense)	(None, 1)	33

Total params: 447,969
Trainable params: 447,969
Non-trainable params: 0

Figure 4. 8: GRU architecture

While modelling GRU neural network, epochs is selected as 20, and batch_size and validation_data remains the same as in our previous neural network models. Our GRU model provides us predictions presented in Table 4.13 below

Accuracy Score: 0.5848			
	Precision	Recall	F1 Score
Fake (0)	0.61	0.68	0.64
Real (1)	0.54	0.47	0.50

Table 4. 14: GRU Result

Bi-directional LSTM

For our Bi-directional LSTM architecture, we are using 5 hidden layers for better predictions. Units for each hidden layer is selected as 32, and dropout as 0.2. The activation function for hidden and output layers remains the same as 'ReLU' and 'sigmoid'. Optimizer and loss function are also like other models. Our Bi-Directional LSTM model summary is given in Figure 4.9

In our modelling of Bi-Directional LSTM architecture, 20 epochs again provided best predictions, and predictions are converted with similar procedure into numeric 0 and 1. Our result from Bi-Directional LSTM is provided on the next page in Table 4.13

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_40 (Embedding)     (None, 20, 32)            404256
_________________________________________________________________
bidirectional_58 (Bidirectio (None, 20, 64)            16640
_________________________________________________________________
bidirectional_59 (Bidirectio (None, 20, 64)            24832
_________________________________________________________________
bidirectional_60 (Bidirectio (None, 20, 64)            24832
_________________________________________________________________
bidirectional_61 (Bidirectio (None, 20, 64)            24832
_________________________________________________________________
bidirectional_62 (Bidirectio (None, 20, 64)            24832
_________________________________________________________________
bidirectional_63 (Bidirectio (None, 64)                24832
_________________________________________________________________
dense_99 (Dense)             (None, 1)                 65
=================================================================
Total params: 545,121
Trainable params: 545,121
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. 9: Bi-Directional LSTM architecture

Accuracy Score: 0.5538			
	Precision	Recall	F1 Score
Fake (0)	0.58	0.69	0.63
Real (1)	0.50	0.39	0.44

Table 4. 15: Bi-Directional LSTM Result

4.6 Comparison of Machine Learning and Deep Learning Models

For the comparison of our fake news detection models using machine learning and deep learning methods, we will be using performance metrics of 'Accuracy' and 'Recall' for news label 'Fake' among four performance metrics of accuracy, precision, recall, and F1 score. Below given Table 4.15 is the tabular representation of our collected results:

Fake News Detection Models	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.5198	0.47	0.58	0.52
Random Forest	0.6029	0.56	0.71	0.59
XGBoost	0.5735	0.51	0.68	0.58
Naïve Bayes	0.5695	0.51	0.68	0.58
Logistic Regression	0.5924	0.53	0.73	0.62
Support Vector Machine (SVM)	0.5834	0.52	0.76	0.62
Linear SVC	0.5685	0.51	0.64	0.57
Artificial Neural Network (ANN)	0.5588	0.6	0.59	0.59
Convolution Neural Network (CNN)	0.5658	0.6	0.64	0.62
Recurrent Neural Network (RNN)	0.5578	0.61	0.57	0.59
Long short-term Memory (LSTM)	0.5858	0.62	0.66	0.64
Gated Reccurent Units (GRU)	0.5848	0.61	0.68	0.64
Bi-Directional LSTM	0.5538	0.58	0.69	0.63

Table 4. 16 Fake News Detection Models on Performance Metrics

We have discussed in section 3.4 that 'Accuracy' is a ratio of total correctly predicted values to the total number of values. Therefore, accuracy for our models would mean ability to correctly predict the label of news i.e., either 'Real' or 'Fake.' Accuracy of our models is presented in the graphical analysis in below given Figure 4.10.

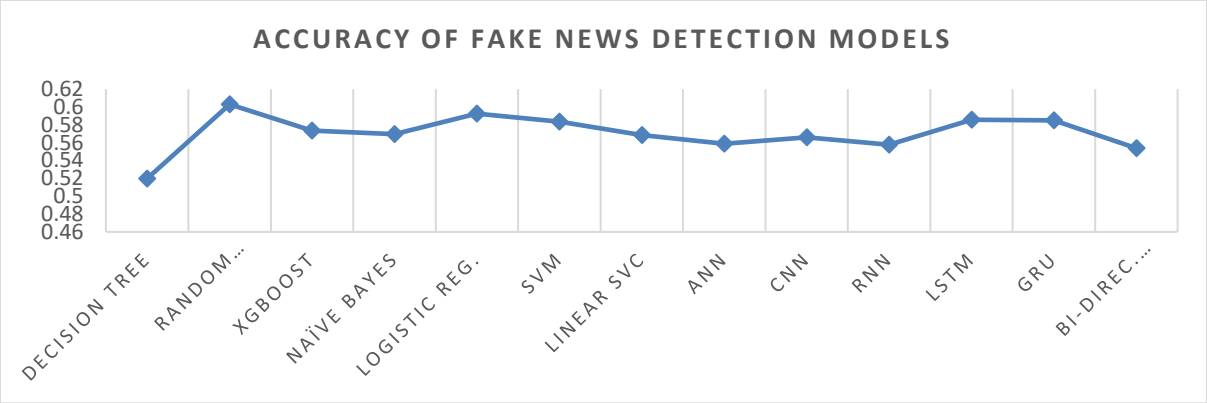


Figure 4. 10 Fake News Detection Model Accuracy

Similarly, 'Recall' is the ratio of total number of correctly predicted positive values to the actual number of positive values.

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

In case of fake news, True Positive means prediction is 'fake' and it is accurately predicted as news is 'fake'. False Negative means prediction is 'real' and it is inaccurately predicted as news is 'fake'. Recall would thus mean, 'Of all the news which are fake, how many of those we correctly predicted fake?' Therefore, of all the performance metrics for 'Fake' news label, we will be using 'Recall' metrics to understand which model performed better.

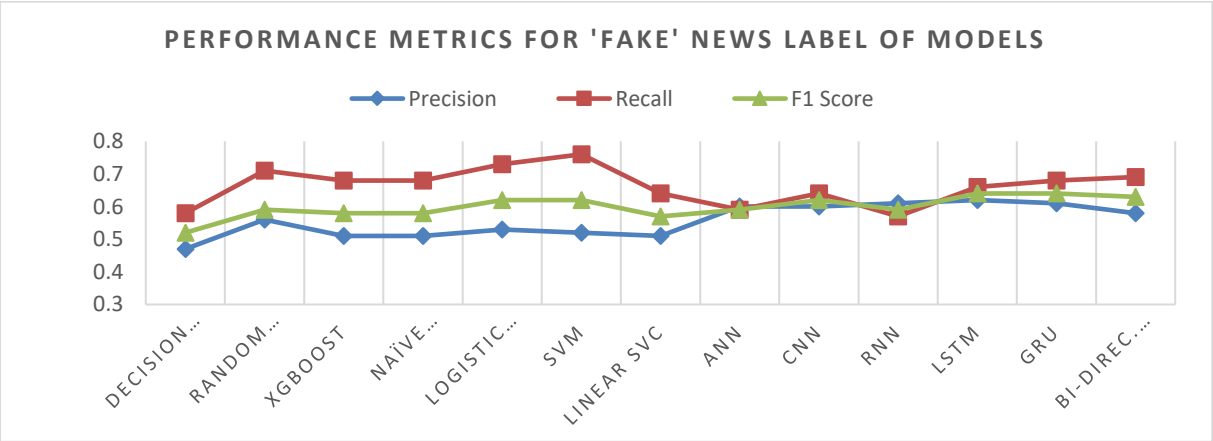


Figure 4. 11 Performance Metrics for Fake News Label of Models

Figure 4.11 given on the previous page is graphical analysis of performance metrics of our fake news detection models. Next, we will now plot our model performance in a scatterplot with x-axis as 'Accuracy' and y-axis as 'Recall' to finally draw conclusion on the performances of our fake news detection models.

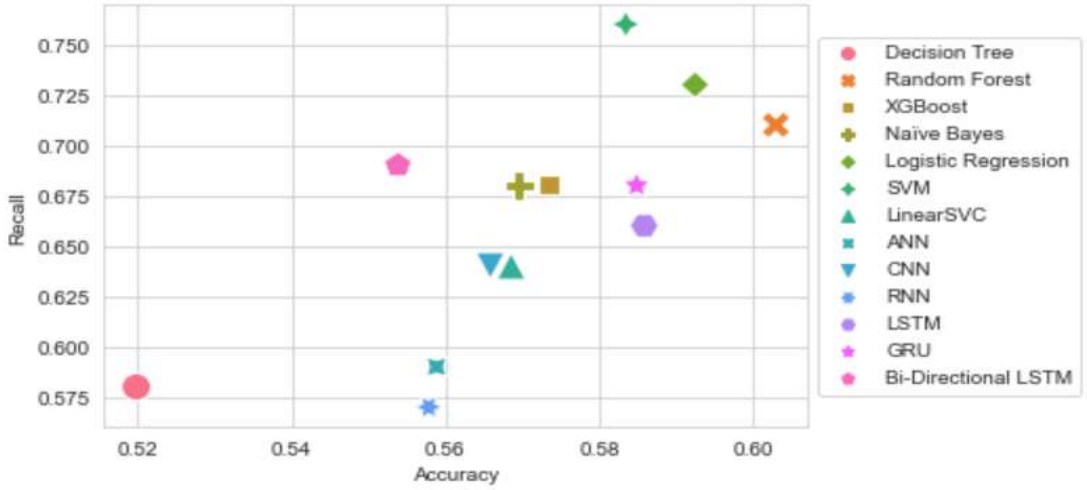


Figure 4. 12 Models on Accuracy and Recall (Fake News Label): A scatterplot

Above given Figure 4.12 is the scatterplot of our thirteen fake news detection models and our conclusion is presented in Table 4.16 below:

	Top Performing Models	Poor Performing Models
Machine Learning	Logistic Regression, SVM, Random Forest	Decision Tree, LinearSVC
Deep Learning	LSTM, GRU	ANN, RNN

Table 4. 17 Result of Comparative Study

In this chapter, we have conducted our research work for fake news detection. We started with collection of our LIAR dataset, and then proceeded with data pre-processing. After that, we performed feature extraction with NLP technique on text data. Next, we implemented our machine learning models with these algorithms: Decision Tree, Random Forest, XGBoost, Naïve Bayes, Logistic Regression, Support Vector Machine, and Linear SVC. After that, we implemented six neural networks to conduct our study of fake news detection. Neural network implemented models are ANN, CNN, RNN, LSTM, GRU and Bi-Directional LSTM. Eventually, we compared our gathered results from each of thirteen fake news detection models on their performances metrics and presented our conclusion with top and poor performing fake news detection methods.

5 Evaluation and Reflection

We have conducted our research study for the fake news detection problem in the previous chapter with 13 different models, and present our result with Logistic Regression, Support Vector Machine, Random Forest, LSTM and GRU as top performing model, and Decision Tree, LinearSVC, ANN and RNN as poor performing model. In this chapter, we will be evaluating the rationale behind performances of each model and our methods of implementing them. We will also be reflecting on our approach in the last section.

5.1 Evaluation

Aim of our study is to conduct comparative analysis of existing supervised learning methods for fake news detection. We have chosen the LIAR dataset among Buzzfeed, Kaggle, UNBiased, News Trend, Reuters, and LIAR datasets identified in literature review. This dataset has 12.8 manually labelled columns proportionally divided into training set, test set, and validation set of size 10269, 1283, 1284, respectively. Even though, in data processing we dropped 'Half-True' label columns and shortened six labels to two i.e, Real and Fake, our dataset remained with more than 10,000 columns and well-balanced. Modelling on this dataset has made our results reliable, and our results are also supported by other comparative studies of few machine learning algorithms.

In our study**,** we have implemented seven machine learning algorithms namely Decision Tree, Random Forest, Naïve Bayes, XGBoost, LinearSVC, SVM, Logistic Regression on our LIAR dataset where Decision Tree, LinearSVC performed poorly and Random Forest, SVM, and Logistic Regression were top performer**.**

Decision Tree failed to predict with more accuracy because it learns to predict with a set of if-then-else decision rules at each node. Being greedy algorithm, at any given node, if leaves have similar features, Decision Tree greedily picks the best one with strong features. This creates the problem of high correlation and overfitting, leaving Decision Tree with least impressive scores. Random Forest, on the other hand, is an ensemble of Decision Tree, and created to solve this problem of overfitting in Decision Tree. In Random Forest, we use bootstrap aggregation of *n* number of

decision trees, and at each node a random sample of *m* features are selected from the full set of features, *p* where usually $m = \sqrt{p}$. A fresh sample is taken at each node, and thus making not just strong, but all features participate. This helped Random Forest to not only decorrelate trees but make predictions with one of the best scores in our models. In our modelling of Decision Tree and Random Forest, we have chosen Gini index as a measure of quality of split with maximum depth for leaves until gini = 0.0. We have also used all features when looking for the best split at intermediary leaves in Decision Tree and chosen square root of the total number of features as maximum feature in Random Forest. Key thing in modelling fake news detection model with Decision Tree and Random Forest is to not limit the depth of leaves. If tree is allowed to go until leaves become pure i.e, gini = 0.0, feature extracted from the text of news columns would be fully utilized for better prediction.

LinearSVC and Support Vector Machine (SVM) are another set of similar algorithms used in our study. In both algorithms, the aim is to generate *n-1* hyperplane in *n* dimensional space with the aim of maximising the margin between output labels. LinearSVC, however, uses linear kernel for modelling which produces decision boundary for hyperplane as a straight line. It makes the predicted output highly unbalanced. SVM, on the other hand, uses Gaussian Kernel which improves its hyperplane and helps to predict a balanced predicted output. Based on this reason, SVM performed well in our modelling, and despite being similar to SVM, LinearSVC performed poorly with 64% on recall score, and 56% at accuracy. In our modelling of LinearSVC, default values of Scikit-Learn such as class weight 1, and loss function as 'squared hinge' were suitable, but we changed C=10 to help decrease regularization and increase performance. If we were making predictions on all six-news labels of the LIAR dataset, we would be using 'hinge' as loss function. Squared hinge loss function is more suitable for binary classification. While modelling our model on SVM, the default kernel in Scikit-Learn, Radical Basis Function (RBF) is the most suitable to establish a decision boundary with maximum margin. The most important thing with SVM modelling is to find the best values for C and gamma parameter of the kernel. We obtained best result in our modelling with C=10, and gamma with its default of 'scale' in Scikit-Learn, which is equal to 1/ (n_features * X.var()) despite the fact that gridsearch suggested C=0.1 and gamma=1.

Logistic Regression is completely different from other algorithms used in our study as instead of predicting output labels directly, it predicts the probability of output labels ranging between 0 and 1 with sigmoid or logistic function. This helps the Logistic Regression model outperform other models. In our modelling through scikit-Learn, solver 'lbfgs' was used to minimise loss function, and C=1 used for optimal regularization. Both output class 0 and 1 were given equal weight so that we have the news label 'Real' if $p(Y) > 0.5$. Lowering the threshold than $p(Y) > 0.5$ would have made our predictions biased towards 'fake' news label, and our results with Logistic Regression modelling as 59% accuracy on news labels and 73% recall on fake news label is thus reliable.

We have also conducted, in our study, fake news detection models on deep learning methods namely ANN, RNN, CNN, LSTM, GRU and Bi-Directional LSTM. ANN is one of the simplest of neural networks. In ANN, based on weight input gets passed into the hidden layer, where an activation function decides whether to pass information to the next layer or not. ANN does not have any other feature to significantly improve each passing epoch. This causes ANN to improve very little on each hidden layer. RNN, on the other hand, is built over ANN and much improved architecture. In RNN, hidden layers are grouped together, and it gets input from the neuron at the previous time step as well. Ideally, it should help improve prediction significantly and RNN should outperform ANN with a good margin, but with the problem of vanishing stochastic gradient, it fails to improve after few epochs as all nodes share the same weight. Based on these reasons, ANN and RNN performed poorly in our fake news detection model.

The important thing with our modelling of deep learning methods is to pick an appropriate number of neurons in each layer, hidden layer, activation function, optimiser and loss function. Optimiser is critical to update weight with learning rate. Among implemented optimiser of RMSprop, Adam, AdaGrad in other studies of fake news detection, Adam optimiser in our implementation provided best results. Learning rate is suggested as the most important hyperparameter to tune in deep learning (Goodfellow, Bengio and Courville, 2016). Hence, for each of the models, we carefully picked the learning rate for Adam optimiser from default of 1e-2 to 1e-5. Loss function is another critical selection, and for binary classification purpose of our

model, binary_crossentropy proved better in comparison with squared hinge, and hinge loss function. Models then tried and tested with different number of epochs and batch size until loss on validation set stops decreasing before making predictions.

In all our deep learning methods for fake news detection, LSTM and GRU performed well. Both neural networks are very similar in architecture and built to overcome the shortcoming in RNN – the vanishing gradient descent. LSTM architecture has a memory cell and three gates on every neuron to continuously improve on weight. This way the model does not get stuck, and each passing epoch learns to make better prediction. GRU architecture has no memory cell but update and reset gates to keep improving weights. Along with these features, another critical thing that helped LSTM and GRU improve in prediction and outperform other models is dropout regularization function. We carefully picked the dropout rate as 0.1 and 0.2 for LSTM and GRU, respectively. This meant 10-20% of neurons were ignored while forward propagation, and backpropagation each passing epoch. This helped LSTM and GRU prevent overfitting while continuously improve with each passing epoch, making them top performing deep learning model in our study.

5.2 Reflection

Fake news is one of the major threats to informed decision making of people, and it has evoked a substantial amount of research work. Aim of this study was, therefore, to conduct a comparative analysis of existing supervised learning methods for fake news detection. From the literature review in chapter 2, we have identified thirteen methods, For our study, thereafter, a dataset with more than 10,000 columns was required to overcome the problem of under sampling and get reliable results. We have identified two datasets from literature review namely 'Liar, Liar, Pants on Fire' (LIAR) dataset with 12.8k columns, and Reuters with 19.9K suitable columns. A comparative study on Reuters dataset presented impressive results like 94% for Logistic Regression and 84% for Decision Tree (Kaur, Kumar and Kumaraguru, 2019). However, we decided on the LIAR dataset as it has six news labels, and we assumed that statements under news labels like 'Barely-True', and 'Mostly-True' could be a fitting test for an algorithm for fake news detection task. Afterwards, we

modelled our machine learning and deep learning methods as detailed separately in Jackson Structured Programming graph below in Figure 5.1

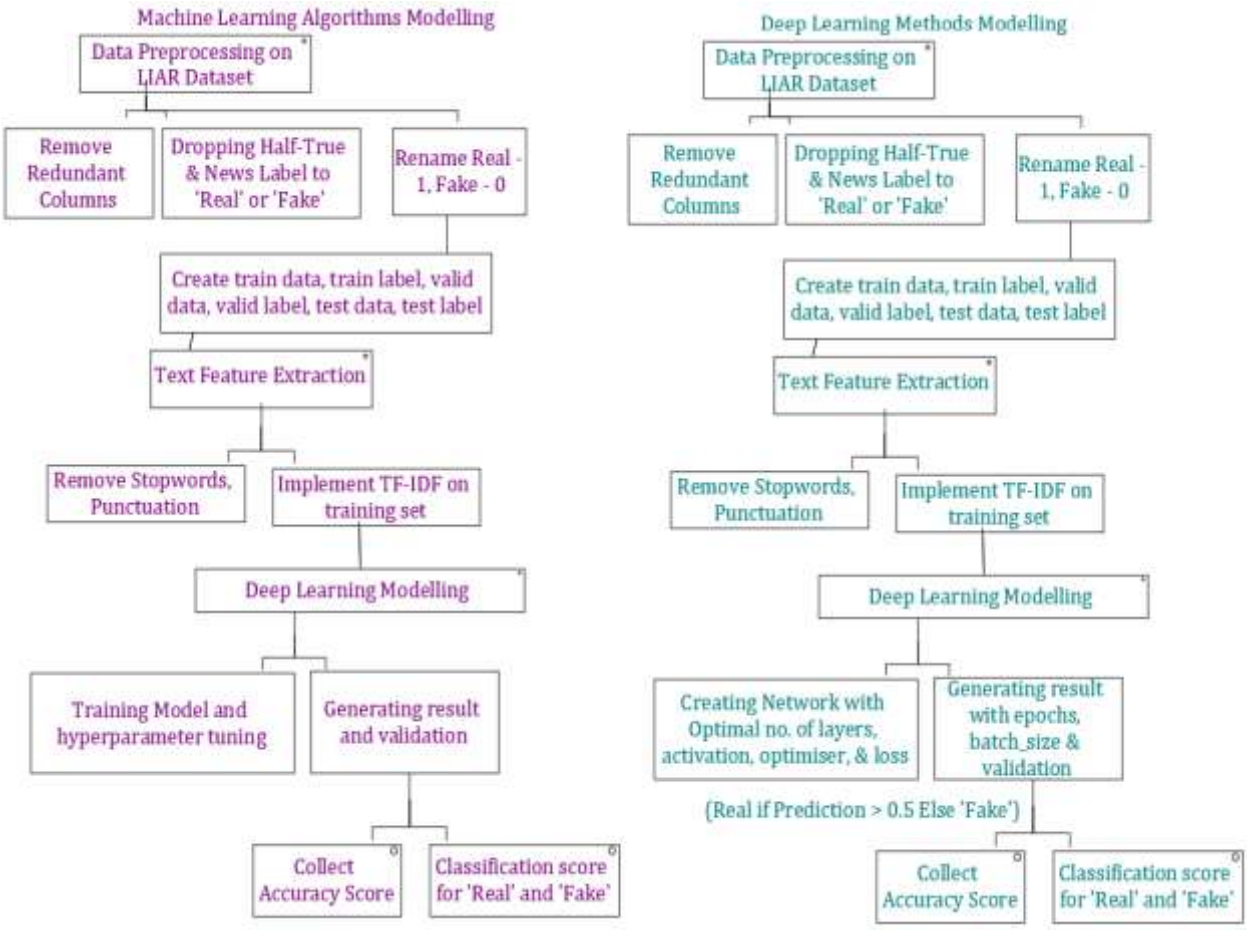


Figure 5. 1 Jackson Structured Programming of Machine Leaning and Deep Learning Fake News Detection Models

In our study, each machine learning and deep learning model has been reworked with hyperparameter tuning and regularization to bring out best possible predictions, and we have obtained Accuracy score ranging from 51% to 60% and Recall from 58% to 76%. Random Forest, SVM and Logistic Regression are our top performing machine learning algorithms, and our result is supported by other comparative study on machine learning algorithms. We probably could have improved on Accuracy and Recall if latest NLP technique like Word2Vec or Hashing Vectorizer were implemented. Implementation on Reuters dataset and Ensemble methods too would have helped improve Accuracy and Recall. Also, if we would have conducted our study on two or more datasets, we could have used post-hoc Nemenyi Test to firmly assert our findings. However, our results from fake news detection

models on LIAR dataset can be considered reliable, and another study (Agarwal, Sultana, Malhotra, and Sarkar, 2019) on LIAR dataset supported our results. We have slightly improved Recall score from (Agarwal, Sultana, Malhotra, and Sarkar, 2019) study probably because of our different approach to news label categorization, and hyperparameter tuning. This has made our model to be able to detect fake news moderately better. Difference in our study and (Agarwal, Sultana, Malhotra, and Sarkar, 2019) study can be seen below in Figure 5.2.

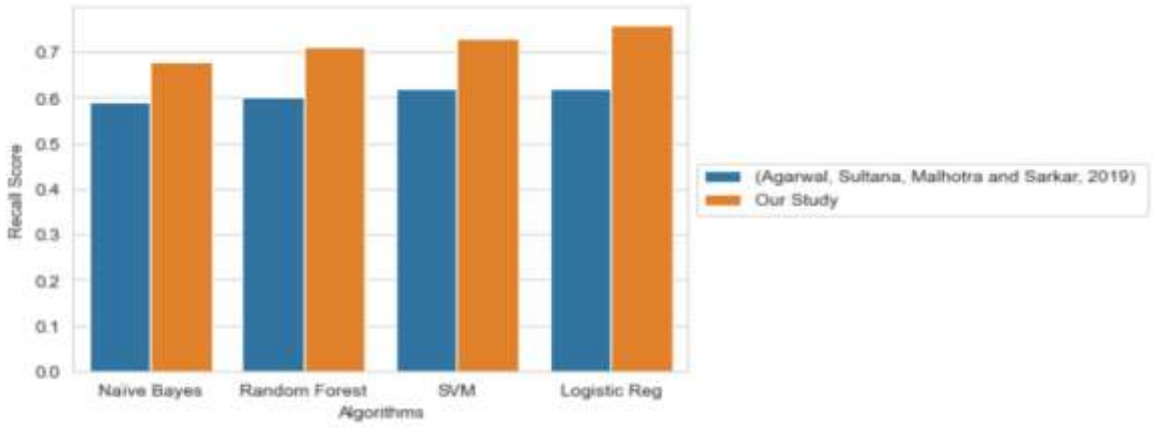


Figure 5. 2 Comparison of Recall with another study on LIAR dataset

All objectives set out for the successful completion of the aim of our study has been fulfilled through this work and we presented Logistic Regression, Support Vector Machine, Random Forest as our top performing machine learning algorithms and LSTM and GRU as our best performing deep learning methods. Deep learning methods takes longer to train on neural network, so for quicker solution to fake news detection, machine learning algorithms could be a suitable choice.

6 Proposed Future Study

We have conducted this study on 13 different models ranging from machine learning decision tree to GRU neural network for fake news detection problem and found that the machine learning algorithms like Random Forest, Logistic Regression and Support Vector Machine (SVM) and deep learning methods like LSTM and GRU are more efficient for the task of fake news detection.

We expect, in future, more research work to be conducted with Logistic Regression and SVM. Also, LSTM and GRU architecture is efficient and can be utilized in further research work as well. On the other hand, newer word embedding techniques such as Word2Vec and Hashing Vectorizer is promising and could be implemented with machine learning and neural networks for further research.

Word2Vec: It is a word embedding technique where each word is represented as a real-valued vector in a vector space of several hundred dimensions. Words with similar meaning have similar representation in that vector space of multi-dimensional mode. Word2Vec has two model architectures, first to predict the current word from the surrounding context words, and second architecture uses current word to predict surrounding context words.

Hashing Vectorizer: With hashing vectorizer, there is no need to store a vocabulary dictionary (Scikit-learn.org, 2020). Each token for a word directly maps to a column position in Document Term Matrix, which reduces memory requirement to much lower level than Word2Vec or TF-IDF. Therefore, it is more suitable when working with dataset of bigger size.

For future improvements, first, we recommend a bigger size of the dataset like Reuters dataset, if studies are not being conducted on LIAR dataset. If one would train their model on a bigger training set, they should expect better and reliable results. For smaller dataset, Word2Vec would be a suitable NLP choice than TF-IDF. Hashing Vectorizer is more suitable to work on datasets bigger than Reuters. Also, studies can be conducted on two or more datasets to assert on findings more firmly. Lastly, implementation with algorithms like Random Forest, Logistic Regression and SVM or deep learning methods like LSTM and GRU for better results.

Bibliography

Agarwal, V., Sultana, H., Malhotra, S. and Sarkar, A., 2019. Analysis of Classifiers for Fake News Detection. *Procedia Computer Science*, 165, pp.377-383.

Ahmed, H., Traore, I. and Saad, S., 2017, October. Detection of online fake news using n-gram analysis and machine learning techniques. In *International conference on intelligent, secure, and dependable systems in distributed and cloud environments* pp. 127-138. Springer, Cham.

Allcott, H. and Gentzkow, M., 2017. Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives*, 31(2), pp.211-236.

Alpaydın, E., 2014. *Introduction To Machine Learning*. 3rd ed. MIT Press.

Bahad, P., Saxena, P. and Kamal, R., 2019. Fake News Detection using Bi-directional LSTM-Recurrent Neural Network. *Procedia Computer Science*, 165, pp.74-82.

Bakir, V. and McStay, A., 2017. Fake News and The Economy of Emotions. *Digital Journalism*, 6(2), pp.154-175.

Barua, R., Maity, R., Minj, D., Barua, T. and Layek, A., 2019. F-NAD: An Application for Fake News Article Detection using Machine Learning Techniques. *2019 IEEE Bombay Section Signature Conference (IBSSC)*. pp. 1-6.

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

Bolton, D.M. and Yaxley, J., 2017. Fake news and clickbait–natural enemies of evidence-based medicine. *BJU international*, *119*, pp.8-9.

Bottou, L., 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of COMPSTAT'2010*, pp.177-186.

Breiman, L., 2001. Random forests. *Machine learning*, *45*(1), pp.5-32.

Brennen, J.S., Simon, F., Howard, P.N. and Nielsen, R.K., 2020. Types, sources, and claims of COVID-19 misinformation. *Reuters Institute*, 7, pp.3-1.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dey, R. and Salem, F., 2017. Gate-variants of Gated Recurrent Unit (GRU) neural networks. *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. Boston, MA, 2017, pp. 1597-1600

FactCheck.org. 2020. *Factcheck.Org*. [online] Available at: https://www.factcheck.org/ [Accessed 19 August 2020].

Factmata.com. 2020. *Factmata*. [online] Available at: https://factmata.com/ [Accessed 19 August 2020].

Fang, Y., Gao, J., Huang, C., Peng, H. and Wu, R., 2019. Self Multi-Head Attention-based Convolutional Neural Networks for fake news detection. *PLOS ONE*, 14(9), p.e0222713.

Faustini, P. and Covões, T., 2020. Fake news detection in multiple platforms and languages. *Expert Systems with Applications*, 158, p.113503.

Gartner. 2020. *Gartner Reveals Top Predictions For IT Organizations And Users In 2018 And Beyond*. [online] Available at: https://www.gartner.com/en/newsroom/press-releases/2017-10-03-gartner-reveals-top-predictions-for-it-organizations-and-users-in-2018-and-beyond [Accessed 19 August 2020].

Ghafari, S., Yakhchi, S., Beheshti, A. and Orgun, M., 2018. Social Context-Aware Trust Prediction: Methods for Identifying Fake News. *Web Information Systems Engineering – WISE 2018*, pp.161-177.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep learning*. The MIT Press, p.429.

Granik, M. and Mesyura, V., 2017, May. Fake news detection using naive Bayes classifier. In *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)* pp. 900-903.

Gravanis, G., Vakali, A., Diamantaras, K. and Karadais, P., 2019. Behind the cues: A benchmarking study for fake news detection. *Expert Systems with Applications*, 128, pp.201-213.

Greff, K., Srivastava, R., Koutnik, J., Steunebrink, B. and Schmidhuber, J., 2017. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10), pp.2222-2232.

Hargreaves, S., 2020. *Steve Jobs Rumor Causes Brief Fall In Apple Stock - Oct. 3, 2008*. [online] Money.cnn.com. Available at: https://money.cnn.com/2008/10/03/technology/apple/ [Accessed 19 August 2020].

Jadhav, S. and Thepade, S., 2019. Fake News Identification and Classification Using DSSM and Improved Recurrent Neural Network Classifier. *Applied Artificial Intelligence*, 33(12), pp.1058-1068.

Jang, B., Kim, I. and Kim, J., 2019. Word2vec convolutional neural networks for classification of news articles and tweets. *PLOS ONE*, 14(8), p.e0220976.

Kaliyar, R., Goswami, A., Narang, P. and Sinha, S., 2020. FNDNet – A deep convolutional neural network for fake news detection. *Cognitive Systems Research*, 61, pp.32-44.

Kaliyar, R., Goswami, A. and Narang, P., 2020. MCNNet: Generalizing Fake News Detection with a Multichannel Convolutional Neural Network using a Novel COVID-19 Dataset. *8th ACM IKDD CODS and 26th COMAD*.

Kaur, S., Kumar, P. and Kumaraguru, P., 2019. Automating fake news detection system using multi-level voting model. *Soft Computing*, 24(12), pp.9049-9069.

Kesarwani, A., Chauhan, S., Nair, A. and Verma, G., 2020. Supervised Machine Learning Algorithms for Fake News Detection. *Lecture Notes in Electrical Engineering*, pp.767-778.

Khattar, D., Goud, J.S., Gupta, M. and Varma, V., 2019, May. Mvae: Multimodal variational autoencoder for fake news detection. In *The World Wide Web Conference*. pp. 2915-2921.

Kim, K. and Jeong, C., 2019. Fake News Detection System using Article Abstraction. *16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 209-212.

Ko, H., Hong, J., Kim, S., Mesicek, L. and Na, I., 2019. Human-machine interaction: A case study on fake news detection using a backtracking based on a cognitive system. *Cognitive Systems Research*, 55, pp.77-81.

Kuo, C.C.J., 2016. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, *41*, pp.406-413.

Kwon, S., Cha, M. and Jung, K., 2017. Rumor Detection over Varying Time Windows. *PLOS ONE*, 12(1), p.e0168344.

Larose, D.T., 2006. Naïve Bayes Estimation and Bayesian Networks. In *Data Mining Methods and Models*. Hoboken, NJ, USA: John Wiley & Sons, Inc, pp. 204–239.

Larose, D.T., 2006. Logistic Regression. In *Data Mining Methods and Models*. Hoboken, NJ, USA: John Wiley & Sons, pp. 155–203.

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *Nature*, 521(7553), pp.436-444.

Lewandowsky, S., Ecker, U., Seifert, C., Schwarz, N. and Cook, J., 2012. Misinformation and Its Correction. *Psychological Science in the Public Interest*, 13(3), pp.106-131.

McTear M., Callejas Z., Griol D. 2016. Spoken Language Understanding. In *The Conversational Interface*. Springer, Cham, pp 161-185

Mills, A. and Robson, K., 2019. Brand management in the era of fake news: narrative response as a strategy to insulate brand value. *Journal of Product & Brand Management*, 29(2), pp.159-167.

Nlp.stanford.edu. 2020. *Stemming And Lemmatization*. [online] Available at: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html [Accessed 19 August 2020].

Nlp.stanford.edu. 2020. *Tokenization*. [online] Available at: https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html [Accessed 19 August 2020].

Pascanu, R., Mikolov, T. and Bengio, Y., 2013, February. On the difficulty of training recurrent neural networks. In *International conference on machine learning* pp. 1310-1318.

Pennycook, G., McPhetres, J., Zhang, Y., Lu, J. and Rand, D., 2020. Fighting COVID-19 Misinformation on Social Media: Experimental Evidence for a Scalable Accuracy-Nudge Intervention. *Psychological Science*, 31(7), pp.770-780.

Pérez-Rosas, V., Kleinberg, B., Lefevre, A. and Mihalcea, R., 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Poddar, K., Amali D., G. and Umadevi, K., 2019. Comparison of Various Machine Learning Models for Accurate Detection of Fake News. *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)*,.

Politifact.com. 2020. *Politifact*. [online] Available at: https://www.politifact.com/ [Accessed 19 August 2020].

Popat, K., Mukherjee, S., Yates, A. and Weikum, G., 2018. Declare: Debunking fake news and false claims using evidence-aware deep learning. *arXiv preprint arXiv:1809.06416*.

Qiu, D., Jiang, H. and Chen, S., 2020. Fuzzy Information Retrieval Based on Continuous Bag-of-Words Model. *Symmetry*, 12(2), p.225.

Reis, J., Correia, A., Murai, F., Veloso, A., Benevenuto, F. and Cambria, E., 2019. Supervised Learning for Fake News Detection. *IEEE Intelligent Systems*, 34(2), pp.76-81.

Rose, J., 2017. Brexit, Trump, and Post-Truth Politics. *Public Integrity*, 19(6), pp.555-558.

Roy, A., Basak, K., Ekbal, A. and Bhattacharyya, P., 2018. A deep ensemble framework for fake news detection and classification. *arXiv preprint arXiv:1811.04670*.

Scherer, D., Müller, A. and Behnke, S., 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. *Artificial Neural Networks – ICANN 2010*, pp.92-101.

Schuster, M. and Paliwal, K., 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), pp.2673-2681.

Scikit-learn.org. 2020. *Sklearn.Feature_Extraction.Text.Tfidfvectorizer — Scikit-Learn 0.23.2 Documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html [Accessed 19 August 2020].

Scikit-learn.org. 2020. *Sklearn.Feature_Extraction.Text.Hashingvectorizer — Scikit-Learn 0.24.0 Documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVectorizer.html [Accessed 20 December 2020].

Shalev-Shwartz, S. and Ben-David, S., 2014 Support Vector Machines. In *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, pp. 167–178.

Shalev-Shwartz, S. and Ben-David, S., 2014 Decision Trees. In *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, pp. 212–218.

Shu, K., Wang, S. and Liu, H., 2019. Beyond News Contents. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 33(1)

Singhal, S. Shah, R., Chakraborty, T., Kumaraguru, P., and Satoh, S. 2019. SpotFake: A Multi-modal Framework for Fake News Detection. *IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, Singapore, Singapore, 2019, pp. 39-47.

Snopes.com. 2020. *Snopes.Com*. [online] Available at: https://www.snopes.com/ [Accessed 19 August 2020].

Swartz, J., 2020. [online] usatoday.com. Available at: https://eu.usatoday.com/story/tech/news/2017/03/11/world-wide-webs-inventor-warns-s-peril/99005906/ [Accessed 19 August 2020].

Team.inria.fr. 2020. *Contentcheck – CEDAR: Rich Data Analytics At Cloud Scale*. [online] Available at: https://team.inria.fr/cedar/contentcheck/ [Accessed 19 August 2020].

Tschiatschek, S., Singla, A., Gomez Rodriguez, M., Merchant, A. and Krause, A., 2018. Fake News Detection in Social Networks via Crowd Signals. *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*. pp 517-524.

Van der Linden, S., Leiserowitz, A., Rosenthal, S. and Maibeach, E. 2017 Incoculating the public against misinformation and climate change. *Global Challenges*. 1(2). p. 1600008.

Watt, J., Borhani, R. and Katsaggelos, A., 2016. Fundamentals of numerical optimization. In *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge: Cambridge University Press, pp.21-44.

Wang, W., 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. pp. 422-426

Werbos, P., 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), pp.1550-1560.

World Economic Forum. 2020. *Fake News: What It Is, And How To Spot It*. [online] Available at: https://www.weforum.org/agenda/2019/03/fake-news-what-it-is-and-how-to-spot-it/ [Accessed 26 August 2020].

Wu, J., 2017. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, *5*, p.23.

Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z. and Yu, P.S., 2018. TI-CNN: Convolutional neural networks for fake news detection. *arXiv preprint arXiv:1806.00749*.

Zannettou, S., Sirivianos, M., Blackburn, J. and Kourtellis, N., 2019. The Web of False Information. *Journal of Data and Information Quality*, 11(3), pp.1-37.

Zhang, J., Dong, B. and Philip, S.Y., 2019, December. Deep Diffusive Neural Network based Fake News Detection from Heterogeneous Social Networks. In *2019 IEEE International Conference on Big Data (Big Data)*. pp. 1259-1266.

Zhang, C., Gupta, A., Kauten, C., Deokar, A. and Qin, X., 2019. Detecting fake news for reducing misinformation risks using analytics approaches. *European Journal of Operational Research*, 279(3), pp.1036-1052.

Appendix A. Research Proposal

1. Aim

Comparative analysis of existing supervised learning methods of classifying fake news from real news.

2. Background, Motivation, Relevance – Literature Review

2.1 Background

Nowadays, internet has become integral part of everyone's life. People rely on internet for every sort of information, and decision-making, and their number is growing exponentially (Dale, 2017). However, internet is filled with fake news and/or misinformation. It hampers everything credible and deludes people into thinking otherwise. Van der Linden et al. (2017) stated that even on climate change issue, misinformation deludes people into thinking that it's hoax. On political front, it is quite widespread, US 2016 election is believed to be heavily influenced by fake news (Allcott and Gentzkow, 2017). E-commerce sector is equally influenced, its been studied that 88% of customers believe in product reviews, and 72% of them believe in business reviews (Zhang and Ghorbani, 2017), and fake reviews or misinformation about them is diverting people into inappropriate decision-making.

Fake news or misinformation could be of various types. It could be some fabricated story, conspiracy theory or political propaganda disseminated by political organization or even government to manipulate public opinion. It can also be innocuous satire just for humorous purpose or hoaxes and rumours to deliberately distract people from some critical issues. Through internet, any person or organisation, commercial or political with malicious intent could sow discord or divert public opinion by creating fake news generated through their bots, trolls, or paid platforms (Zannettou et al., 2019)

It, therefore, can be adjudged as one of the most threatening harm of internet era, and thus its extremely crucial to classify fake news and misinformation from the real ones for the betterment of society, and individual's decision making.

2.2 Related Work

Major computing giants are nowadays involved themselves in classifying misinformation from internet. For instance, Google has started an initiative named Factmata (Factmata.com, 2020) which is complete based on Artificial Intelligence (AI), and Machine Learning (ML). Similarly, Facebook has started Facebook Initiative to detect fake news too.

However, one of the best ways to identify fake news is still to cross-verify it with experts. Many third-party fact-checking organisations like Snopes (Snopes.com, 2020) and Factcheck (FactCheck.org, 2020) are doing that. Factcheck has many dedicated sections like Scicheck for science-based news checking, and Health Watch for healthcare related news. However, any organisation like those can be influenced over a period, so a code of ethics – Ponyter (2016) has been established as well to keep check. Then, we have PolitiFact (Politifact.com, 2020) which rates the accuracy of news by elected political analysts. Major problem with this approach is that – it requires human resources, and thus time-consuming, which makes it practically impossible to do it on all existing information from internet.

Researchers, therefore, have been using data analysis algorithms to classify misinformation. Data analysis algorithms have proven useful in this sort of task, and currently there are many relevant researches have been conducted using supervised learning strategy of data analysis (Bondielli and Marcelloni, 2019).

(Rubin et al., 2016) used Support Vector Machine (SVM) to train detection technique with some content-based features and obtained F1 score of 0.87. Similarly, (Singhania, Fernandez and Rao, 2017) used deep learning model on words, sentences, and news headlines and got 96% accuracy.

There are other related works as well with models mostly based on content-based features, and in various types of misinformation. (Shu et al., 2017) used decision tree, and logistic regression data mining techniques to detect fake news from social media. From social media again, (Tschiatschek et al., 2018) detected fake news via crowd signals. They have developed an algorithm DETECTIVE implementing Bayesian technique to flag fake news from given set of expert reviewed news. (Volkova et al., 2017) have used neural networks to classify suspicious news post from twitter with F1 score as high as 0.92. Among latest work on social media, (Ozbay and Alatas, 2020) combined text mining method and supervised learning algorithms and evaluated with 23 different algorithms with good precision to detect fake news.

When it comes to news articles, (Wei and Wan, 2017) developed an approach to identify ambiguous and misleading headlines from real one using Class Sequence Rules (CSR) mining with F1 score of 0.75. (Dong et al., 2018) used Autoencoders and Random Forest (RF) data analysis technique to come up with 95% accuracy. (Ruchansky, Seo and Liu, 2017) used Recurrent Neural Network (RNN) and achieved 89% accuracy for detecting fake news. Similarly, (Tacchini et al., 2017) got 97% accuracy for finding fake news with Logistics regression.

To data, it can be said that there are many researches has been done with multiple algorithms to find fake news namely, Decision Tree, Random Forest, Logistic Regression, SVM, RNN, etc. However, there it seems necessary to present a comparative study of all these algorithms on a similar dataset, evaluate and analysis which one is the most suitable for classifying fake news or misinformation.

2.3 Motivation

Internet has provided a new way to information creation and consumption by people. But people tend to follow only their favourite channels or like-minded people on social media which influence their opinion and decision-making in a biased manner, creating an echo chamber (Bessi, 2016). To disseminate echo

chamber, we need to have established a detection algorithm that could work expeditiously before spreading and over a long period of time.

This research of comparative analysis of different data techniques is just for that. As for classifying fake news, relevant and labelled dataset is required, most existing techniques are based on supervised learning methods of data science. So, in this research, efforts will be made towards bringing up a comparative study of all existing methods of supervised learning on a similar dataset, and then conclude which one of those is the most suitable and how.

As for this research, only publicly available dataset has to be used, and also that needs to be not small ones as Benjamin Political News Dataset which contains only 75 stories to be classified into real, fake or satire news, following datasets can be used:

- Burfoot Satire News Dataset: It contains 4000 real news and 233 satire news samples to train and test detection methods on.
- LIAR: (Wang, 2017) has proposed an ideally new benchmark dataset named LIAR for fake news with 12.8K data size as with anything less than 1000 wouldn't be suitable for data science algorithms to work upon and implement.

2.4 Relevance

Although, many great researches have been done already, topic of tackling misinformation and fake news is far from being solved. It has also been accessed to have ample impact on human decision making. (Lewandowsky et al., 2012) stated that democracy relies on a well-informed populace. Well informed citizens can make rational wise decisions for themselves and their society whereas misinformed civilian would make contrary decisions. So, it seems to be of great interest to work upon, for better impact on human lives, especially when comparative study like this could give better answers to which data analysis technique is most suitable for detecting fake news

Researching student has studied decision-support systems and machine learning in previous term and learnt skills on data analysis tools, visualisation, and algorithms, along with working ability with Python, Jupyter-notebook, SciKit Learn,

Pandas, Numpy, and Seaborn. These skill sets will be of great help while working on research topic such as this – comparative analysis of fake news detection technique of supervised learning

Working on research like this as dissertation will be a great learning experience and would immensely help researching students to improve and sharpen skills on machine learning and data science which might open up manifold of opportunities after graduation.

3. Scope, Objectives and Risk

3.1 Scope and Objective

Aim of this research is to bring a comparative analysis of existing methods of supervised learning for classifying fake news or misinformation from the real authentic news.

Following are the objectives for this research:

1. Complete a thorough literature survey for the existing data analysis and machine learning approaches
   Deliverable: A report about which literature were searched and reviewed
   Evaluation: To be done by supervisor whether sufficient or not

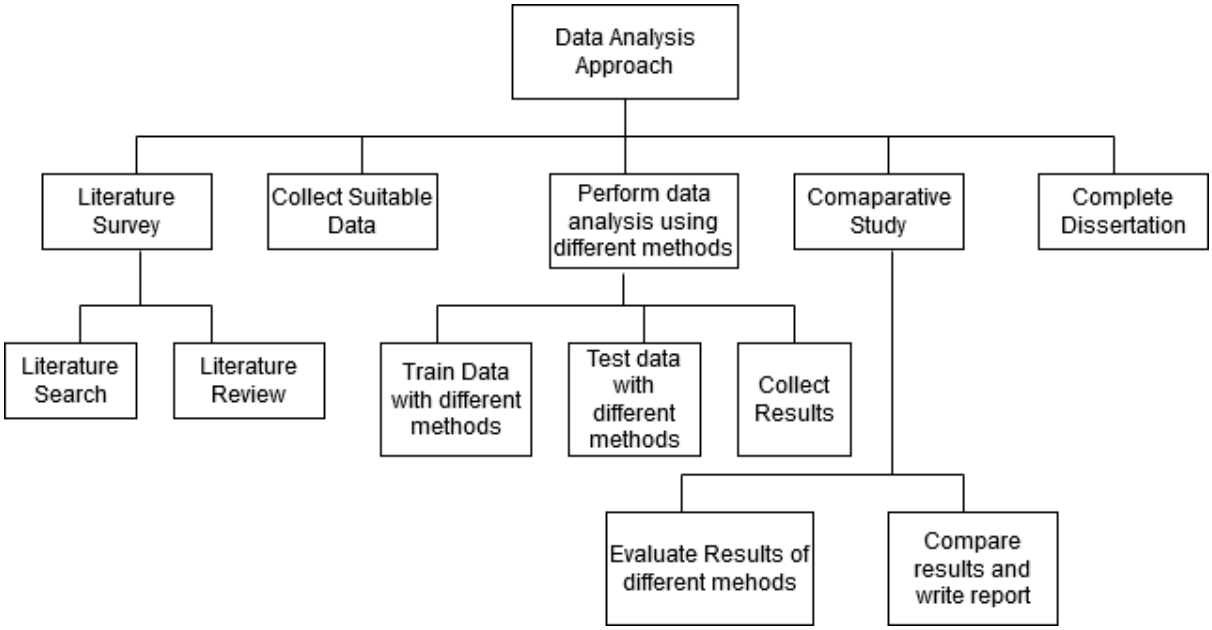
2. Identify and collect suitable data and for analysis and evaluation
   Deliverable: Presenting data in CSV format
   Evaluation: To be done by supervisor whether appropriate or not

3. Perform analysis of each supervised learning method on collected dataset
   Deliverable: A report explaining results from different methods of data analysis
   Evaluation: To be done by supervisor whether efficient or not

4. Compare the final output of different methods using appropriate techniques
   Deliverable: A written report presenting all the result and comparative study
   Evaluation: To be done by supervisor whether accurate or not

5. Complete final report for dissertation
   Deliverable: A written report documenting all work of required word count along with analysis model
   Evaluation: To be done by supervisor whether credible or not.

3.2 Work Breakdown Structure



3.3 Risk Log

Risk Type	Risk Event	Likelihood (1-	Impact (1-10)	Risk Value (1-100)	Risk Monitoring/ Control Flag	Risk Management Strategy	Risk Review Date	Risk Owner
T	Insufficient Literature Review	1	3	10	Delayed Work	Read and take notes. Follow schedule	During Research Period	Researcher

T	Availability of inappropriate data	3	3	30	Delayed Work	Thorough Search through all existing datasets	During Research Period	Researcher
T	Inefficient Analysis of existing models due to lack of deeper understanding	3	4	40	Inaccurate Result/Delayed Work	Read or re-read all the concepts of data analysis	During Research Period	Researcher
F	Financial Risk - None	0	0	0	/	/	/	/
P	Human Resource or Organisation Risk - None	0	0	0	/	/	/	/
E	Environment Risk - None	0	0	0	/	/	/	/
S	Security Risk - None	0	0	0	/	/	/	/

3.4 Task List

Task list is prepared keeping in mind researching student will spend 8 hours a day, 5 days a week Monday to Friday. However, if needed extra hours on weekends will be utilized too.

Task No	Task	Planned Start	Planned End	Task Dependency	Resources Required	Task Outcome	Risk	Comments	Hours
1	Literature Review of existing data analysis and machine learning approaches	17/02/2020	03/03/2020	None	Source of knowledge - Library	Understanding of existing approaches	Important ones on Springer(can get access)	Access to library to be ensured	100
2	Collect Suitable Data	04/03/2020	11/03/2020	Availability of free dataset on internet	Access to internet	Data Collection	Inappropriate and insufficient data	Thorough Search over internet is required	48
3	Perform Analysis using different Methods	12/03/2020	31/03/2020	Deeper understanding of Data Analysis	Source of knowledge – Library and Internet	Analysis result	Inefficient or similar results	To be done n number of times for better results	112

4	Evaluate and compare the analysis result of different methods with Collected Data	20/04/2020	30/04/2020	Suitable dataset, and varying results from different methods	Better Model and Clarified output objective	Comparative Study	Study not being deeply analytical	Deeper understanding and analytical ability required	72
5	Complete Report	01/05/2020	14/05/2020	All previous objectives achieved	Source of knowledge – library	Dissertation Complete	Delayed Work due to previous failures	To be submitted on time	80

4. Source and Use of Knowledge

4.1 Journal

As this research is about data analysis for extracting misinformation or fake news from the internet, it would be appropriate to publish it in Computational Statistics & Data Analysis (CSDA). Most of the relevant papers on data analysis have been published in CSDA. CSDA is one of the prestigious platforms where researchers can publish their papers on statistical and data analysis, and reviewers in CSDA provide high quality reviews to improve upon their work. Also, CSDA provide maximum rights to authors along with intellectual-property protection to their papers

5. Ethics, Legal, Social, Security, and Professional Issues

5.1 Ethics

All the literature resources required for study will be accessed through library, and data to be collected for modelling and analysis will be collected from public

available dataset like Kaggle or Google Dataset. In terms of ethics, there are no issues regarding data confidentiality.

5.2 Legal

For this research only data available in public domain to be utilised, so no legal issue should hinder this research work. Also, as this is individual work, researcher will not involve any other person or plagiarise from internet as per University regulations.

5.3 Social

In terms of social issues, working on this research proposal will not create any problem to society or any individual personally. This research is to carry out individually so no other person or vulnerable groups will be affected.

5.4 Security

As data will be collected from any public domain, this research will not pose any security concern or copyright infringement issue or data theft threats

5.5 Professional

As this is an individual research, and not dependent on other human resource, only researcher's commitment to comply with schedule, standards, and policy is required. All University standard must be followed, and research must be completed with honesty and responsibly.

6.Monitor and Control Table

Objectives	Planned Start Date	Planned End Date	Actual Start Date	Actual End Date	Deliverable	Reflections

Complete a thorough literature survey for the existing data analysis and machine	17/02/2020	03/03/2020			A written report	
Identify and collect suitable data and for analysis and evaluation	04/03/2020	11/03/2020			Data in CSV format	
Perform Data Analysis using different Methods	12/03/2020	31/03/2020			A written report	
Evaluate and compare the analysis result of different methods with Collected Data	20/04/2020	30/04/2020			A written report with comparative study report	
Complete final report for dissertation	01/05/2020	14/05/2020			A final report	

References:

- Allcott, H. & Gentzkow, M. (2017) 'Social Media and Fake News in the 2016 Election' *The Journal of Economic Perspectives*, 31(2), pp.211–235
- Bessi, A. (2016) 'Personality traits and echo chambers on facebook' *Computers in Human Behavior*, 65, pp.319-324
- Bondielli, A. & Marcelloni, F. (2019) 'A survey on fake news and rumour detection techniques', *Information Sciences*, 497, pp.38–55
- Dale, R. (2017) 'NLP in a post-truth world' *Natural Language Engineering*, 23(2), pp.319-324
- Dong, M. Yao, L. Wang, X. Benatallah, B. Huang, C, & Ning, X. (2018) 'Opinion fraud detection via neural autoencoder decision forest' *Pattern Recognition Letters*, pp.

- FactCheck.org. (2020). FactCheck.org. [online] Available at: https://www.factcheck.org/ (Accessed 3 Feb. 2020)
- Factmata.com. (2020). Factmata. [online] Available at: https://factmata.com/ (Accessed 3 Feb. 2020)
- Lewandowsky, S., Ecker, U., Seifert, C., Schwarz, N. and Cook, J. (2012). Misinformation and Its Correction. Psychological Science in the Public Interest, 13(3), pp.106-131
- Rubin, V., Conroy, N., Chen, Y. and Cornwell, S. (2016) 'Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News*'* *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*
- Ruchansky, N., Seo, S. and Liu, Y. (2017) 'CSI: A Hybrid Deep Model for Fake News Detection' *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17*
- Shu, K., Sliva, A., Wang, S., Tang, J. and Liu, H. (2017) 'Fake News Detection on Social Media' *ACM SIGKDD Explorations Newsletter* 19(1), pp.22-36
- Snopes.com. (2020). Snopes.com. [online] Available at: https://www.snopes.com/ (Accessed 3 Feb. 2020)
- Tacchini, E. Ballarin, G. Vedova, M. Moret, S. (2017) 'Some Like it Hoax: Automated Fake News Detection in Social Networks' *Proceedings of the Second Workshop on Data Science for Social Good (SoGood). CEUR Workshop Proceedings* Vol 1960
- The Ponyter Institute (2019) International Fact-Checking Network fact-checkers' code of principles. Available at: https://www.poynter.org/ifcn-fact-checkers-code-of-principles/ (Accessed: 01 Feb. 20)
- Tschiatschek, S. Singla, A. Rodriguez, M.G. Merchant, A. Krause, A. (2018) 'Fake News Detection in Social Networks via Crowd Signals' arXiv:1711.09025v2[cs.S1] Mar 2018
- Van der Linden, S., Leiserowitz, A., Rosenthal, S. and Maibach, E. (2017) 'Inoculating the Public against Misinformation about Climate Change' *Global Challenges*, 1(2), p.1600008
- Volkova, S., Shaffer, K., Jang, J. and Hodas, N. (2017). 'Separating Facts from Fiction: Linguistic Models to Classify Suspicious and Trusted News Posts on Twitter' *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (Volume 2: Short Papers)
- Wang, W. (2017). "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*
- Zannettou, S. Sirivianos, M. Blackburn, J. Kourtellis, N. (2019) 'The Web of False Information: Rumors, Fake News, Hoaxes, Clickbait, and Various Other Shenanigans' *Journal of Data and Information Quality (JDIQ)*, 11(3), pp.1–37
- Zhang, X.A. & Ghorbani, A.A. (2019) 'An overview of online fake news: Characterization, detection, and discussion', *Information Processing and Management*, pp.

Appendix B. List of Figures

Appendix C. List of Tables

Appendix D. Code

THIS SECTION IS FOR MACHINE LEARNING MODEL CODE

```
#!/usr/bin/env python
# coding: utf-8
import numpy as np
import pandas as pd
train_data = pd.read_csv('train.tsv',sep = '\t',
                         names = ['File Type', 'Label', 'Statement', 'Context', 'Speaker',
                                  'Position', 'State', 'Party',
'n1','n2','n3','n4','n5','Source'])
train_data.info()
train_data.head()
train_set = train_data[train_data.Label != 'half-true']
train_setmapping = {'true': 1, 'half-true': 1, 'mostly-true': 1, 'false': 0, 'pants-fire':
0, 'barely-true': 0}
td =train_set.replace({'Label': mapping})
training_set= td.filter(['Label', 'Statement'],axis=1)
training_set
test_data = pd.read_csv('test.tsv', sep = '\t',
                        names = ['File Type', 'Label', 'Statement', 'Context', 'Speaker',
                                  'Position', 'State', 'Party',
'n1','n2','n3','n4','n5','Source'])
test_data.head()
t_set = test_data[train_data.Label != 'half-true']
t_set
mapping = {'true': 1, 'half-true': 1, 'mostly-true': 1, 'false': 0, 'pants-fire': 0,
'barely-true': 0}
td1 = t_set.replace({'Label': mapping})
test_set= td1.filter(['Label', 'Statement'],axis=1)
test_set
valid_data = pd.read_csv('test.tsv', sep = '\t',
                        names = ['File Type', 'Label', 'Statement', 'Context', 'Speaker',
                                  'Position', 'State', 'Party',
'n1','n2','n3','n4','n5','Source'])
valid_data.head()
valid_s = valid_data[train_data.Label != 'half-true']
valid_s
mapping = {'true': 1, 'half-true': 1, 'mostly-true': 1, 'false': 0, 'pants-fire': 0,
'barely-true': 0}
vd = valid_s.replace({'Label': mapping})
valid_set= vd.filter(['Label', 'Statement'],axis=1)
valid_set
X_train = training_set['Statement']
y_train = training_set['Label']
X_test = test_set['Statement']
y_test = test_set['Label']
X_valid = valid_set['Statement']
y_valid = valid_set['Label']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline
txt_clf = Pipeline([('tfidf',TfidfVectorizer()),
('clf',DecisionTreeClassifier(criterion='gini',splitter='best',max_depth=None,max_features=
None)])
txt_clf.fit(X_train,y_train)
predictions = txt_clf.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
#Radom Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
txt_clf = Pipeline([('tfidf',TfidfVectorizer()),
                  ('clf',RandomForestClassifier(n_estimators=150,
    criterion='gini',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features='sqrt'))])
txt_clf.fit(X_train,y_train)
predictions = txt_clf.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
#XGBoost
from xgboost.sklearn import XGBClassifier
from sklearn.pipeline import Pipeline
txt_clf = Pipeline([('tfidf',TfidfVectorizer()),
                  ('clf',XGBClassifier(valid_data=(X_valid, y_valid)))])
txt_clf.fit(X_train,y_train)
predictions = txt_clf.predict(X_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
#NaiveBayes
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
txt_clf = Pipeline([('tfidf',TfidfVectorizer()),
                  ('clf',MultinomialNB())])
txt_clf.fit(X_train,y_train)
predictions = txt_clf.predict(X_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
```

```
#LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
txt_clf = Pipeline([('tfidf',TfidfVectorizer()),
                  ('clf',LogisticRegression(solver='lbfgs',random_state=34,
multi_class="auto", n_jobs=-1, C=1,class_weight="None")])
txt_clf.fit(X_train,y_train)
predictions = txt_clf.predict(X_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
#SupportVectorMachine
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
txt_clf = Pipeline([('tfidf',TfidfVectorizer()),
                  ('clf',SVC(kernel='rbf', C=10, gamma='scale',class_weight=None))])
txt_clf.fit(X_train,y_train)
predictions = txt_clf.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
from sklearn.model_selection import GridSearchCV
param_grid = {'C':[0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001,0.0001]}
grid = GridSearchCV(SVC(),param_grid)
grid.fit(X_train,y_train)
grid.best_params_
grid.best_estimator_
#LinearSVC
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
txt_clf = Pipeline([('tfidf',TfidfVectorizer(stop_words='english')),
                  ('clf',LinearSVC(max_iter=500, C=10))])
txt_clf.fit(X_train,y_train)
predictions = txt_clf.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
```

THIS SECTION IS FOR DEEP LEARNING CODE

```
#!/usr/bin/env python
# coding: utf-8
import tensorflow as tf
print(tf.__version__)
import pandas as pd
import numpy as np
train_data = pd.read_csv('train.tsv',sep = '\t',
                         names = ['File Type', 'Label', 'Statement', 'Context', 'Speaker',
                                  'Position', 'State', 'Party',
'n1','n2','n3','n4','n5','Source'])td= train_data.filter(['Label', 'Statement'],axis=1)
```

```
td
td1 = td[td.Label != 'half-true']
td1
mapping = {'true': 1, 'mostly-true': 1, 'false': 0, 'pants-fire': 0, 'barely-true': 0}
training_set =td1.replace({'Label': mapping})
training_set
# Processing Test Data
test_data = pd.read_csv('test.tsv', sep = '\t',
                        names = ['File Type', 'Label', 'Statement', 'Context', 'Speaker',
                                 'Position', 'State', 'Party',
'n1','n2','n3','n4','n5','Source'])
test_set1= test_data.filter(['Label', 'Statement'],axis=1)
test_set1
tset = test_set1[test_set1.Label != 'half-true']
tset
mapping = {'true': 1, 'mostly-true': 1, 'false': 0, 'pants-fire': 0, 'barely-true': 0}
test_set =tset.replace({'Label': mapping})
test_set
valid_data = pd.read_csv('test.tsv', sep = '\t',
                        names = ['File Type', 'Label', 'Statement', 'Context', 'Speaker',
                                 'Position', 'State', 'Party',
'n1','n2','n3','n4','n5','Source'])
vd= valid_data.filter(['Label', 'Statement'],axis=1)
vd
vd1 = vd[vd.Label != 'half-true']
vd1
mapping = {'true': 1, 'mostly-true': 1, 'false': 0, 'pants-fire': 0, 'barely-true': 0}
valid_set = vd1.replace({'Label': mapping})
valid_set
# Removing punctuation
import string
def remove_punc(text):
    table = str.maketrans("","",string.punctuation)
    return text.translate(table)
training_set['Statement']=training_set['Statement'].map(lambda x: remove_punc(x))
test_set['Statement']=test_set['Statement'].map(lambda x: remove_punc(x))
valid_set['Statement']=valid_set['Statement'].map(lambda x: remove_punc(x))
# remove stopwords
import nltk
from nltk.corpus import stopwords
stop = set(stopwords.words("english"))
def remove_stopwords(text):
    text = [word.lower() for word in text.split() if word.lower() not in stop]
    return " ".join(text)
training_set['Statement']=training_set['Statement'].map(remove_stopwords)
test_set['Statement']=test_set['Statement'].map(remove_stopwords)
valid_set['Statement']=valid_set['Statement'].map(remove_stopwords)
training_set['Statement']
test_set['Statement']
valid_set['Statement']
#counter counts unique words
```

```
#counter to help in allocating num_words
#Creating new set to concatnate all three sets to get exact no. of words in dataset
new_set = pd.concat([training_set, test_set, valid_set], axis=0)
new_set
from collections import Counter
def counter(text):
    count = Counter()
    for i in text.values:
        for word in i.split():
            count[word] +=1
    return count
text = new_set['Statement']
total_count = counter(text)
len(total_count)
num_words = len(total_count)
max_length = 20 #because its says average statement length is 17.9
train_sentences = training_set['Statement']
train_labels = training_set['Label']
test_sentences = test_set['Statement']
test_labels = test_set['Label']
valid_sentences = valid_set['Statement']
valid_labels = valid_set['Label']
from keras.preprocessing.text import Tokenizer
new_set_sentences = new_set['Statement']
tokenizer = Tokenizer(num_words = num_words)
tokenizer.fit_on_texts(new_set_sentences)
word_index = tokenizer.word_index
word_index
train_sequences = tokenizer.texts_to_sequences(train_sentences)
train_sequences[0]
from keras.preprocessing.sequence import pad_sequences
train_padded = pad_sequences(
                train_sequences, maxlen=20, padding="post",truncating = "post")
train_padded[0]
test_sequences = tokenizer.texts_to_sequences(test_sentences)
test_padded = pad_sequences(
                test_sequences, maxlen=20, padding="post",
                truncating = "post")
valid_sequences = tokenizer.texts_to_sequences(valid_sentences)
valid_padded = pad_sequences(valid_sequences, maxlen= 20, padding
="post",truncating="post")
print(f"Shape of the train {train_padded.shape}")
print(f"shape of the test {test_padded.shape}")
# Modelling LSTM
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.initializers import Constant
from keras.optimizers import Adam
model = Sequential()
model.add(Embedding(num_words, 32, input_length=max_length))
model.add(LSTM(32, return_sequences=True, activation="relu", dropout = 0.2))
```

```
model.add(LSTM(32, return_sequences=True, activation="relu", dropout = 0.2))
model.add(LSTM(32, return_sequences=True, activation="relu", dropout = 0.2))
model.add(LSTM(32, return_sequences=True, activation="relu", dropout = 0.2))
model.add(LSTM(32, return_sequences=True, activation="relu", dropout = 0.2))
model.add(LSTM(32, return_sequences=True, activation="relu", dropout = 0.2))
model.add(LSTM(32, activation = "relu", dropout = 0.1))
model.add(Dense(1, activation='sigmoid'))
opt = Adam(learning_rate=0.0001)
model.compile(loss='binary_crossentropy', optimizer=opt,metrics=['accuracy'])
model.summary()
model.fit(train_padded, train_labels, epochs=25, batch_size=32,
validation_data=(valid_padded, valid_labels))
predictions = model.predict(test_padded)
predictions = (predictions > 0.5)
predictions.astype('int64')
from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
print(confusion_matrix(test_labels, predictions))
print(classification_report(test_labels,predictions))
print(accuracy_score(test_labels,predictions))
# Modelling ANN
from keras.layers import Flatten
ann = Sequential()
#First input and hidden layer
ann.add(Embedding(num_words,32, input_length=max_length))
ann.add(Dense(units=32, activation='relu'))
ann.add(Dense(units=32, activation='relu'))
ann.add(Dense(units=32, activation='relu'))
ann.add(Dense(units=32, activation='relu'))
ann.add(Dense(units=32, activation='relu'))
ann.add(Dense(units=32, activation='relu'))
ann.add(Flatten())
#Output layer
ann.add(Dense(units=1))
#compiling
opt = Adam(learning_rate=0.001)
ann.compile(optimizer = opt, loss = 'binary_crossentropy', metrics = ['accuracy'])
ann.summary()
#training ann
ann.fit(train_padded, train_labels, batch_size=32, validation_data=(valid_padded,
valid_labels),epochs=40)
predictions_ann = ann.predict(test_padded)
predictions_ann = (predictions_ann > 0.5)
predictions_ann.astype('int64')
print(confusion_matrix(test_labels, predictions_ann))
print(classification_report(test_labels,predictions_ann))
print(accuracy_score(test_labels,predictions_ann))
# Modelling CNN 1D
from keras.layers import Conv1D, GlobalAveragePooling1D,Dense
cnn = Sequential()
#First input and hidden layer
cnn.add(Embedding(num_words, 32, input_length=max_length))
```

```
cnn.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
cnn.add(GlobalAveragePooling1D())
cnn.add(Dense(24,activation='relu'))
cnn.add(Dense(24,activation='relu'))
#Output layer
cnn.add(Dense(1, activation='sigmoid'))
#compiling
opt = Adam(learning_rate=0.00001)
cnn.compile(optimizer = opt, loss = 'binary_crossentropy', metrics = ['accuracy'])
cnn.summary()
#training cnn
cnn.fit(train_padded, train_labels, epochs=15, batch_size=32,
validation_data=(valid_padded, valid_labels))
predictions_cnn = cnn.predict(test_padded)
predictions_cnn = (predictions_cnn > 0.5)
predictions_cnn.astype('int64')
print(confusion_matrix(test_labels, predictions_cnn))
print(classification_report(test_labels,predictions_cnn))
print(accuracy_score(test_labels,predictions_cnn))
# Modelling GRU
from keras.layers import GRU
gru = Sequential()
gru.add(Embedding(num_words, 32, input_length=max_length))
gru.add(GRU(32, return_sequences=True, activation="relu", recurrent_activation='sigmoid',
dropout = 0.2))
gru.add(GRU(32, return_sequences=True, activation="relu", recurrent_activation='sigmoid',
dropout = 0.2))
gru.add(GRU(32, return_sequences=True, activation="relu", recurrent_activation='sigmoid',
dropout = 0.2))
gru.add(GRU(32, return_sequences=True, activation="relu", recurrent_activation='sigmoid',
dropout = 0.2))
gru.add(GRU(32, return_sequences=True, activation="relu", recurrent_activation='sigmoid',
dropout = 0.2))
gru.add(GRU(32, return_sequences=True, activation="relu", recurrent_activation='sigmoid',
dropout = 0.2))
gru.add(GRU(32, activation = "relu", dropout = 0.1))
gru.add(Dense(1, activation='sigmoid'))
opt = Adam(learning_rate=0.0001)
gru.compile(loss='binary_crossentropy', optimizer=opt,metrics=['accuracy'])
gru.summary()
gru.fit(train_padded, train_labels, epochs=20, batch_size=32,
validation_data=(valid_padded, valid_labels))
predictions_gru = gru.predict(test_padded)
predictions_gru = (predictions_gru > 0.50)
predictions_gru.astype('int64')
print(confusion_matrix(test_labels, predictions_gru))
print(classification_report(test_labels,predictions_gru))
print(accuracy_score(test_labels,predictions_gru))
# Modelling Bi-Directional LSTM
from keras.layers import Bidirectional
bilstm = Sequential()
```

```
bilstm.add(Embedding(num_words, 32, input_length=max_length))
bilstm.add(Bidirectional(LSTM(32, return_sequences=True, activation="relu", dropout =
0.2)))
bilstm.add(Bidirectional(LSTM(32, return_sequences=True, activation="relu", dropout =
0.2)))
bilstm.add(Bidirectional(LSTM(32, return_sequences=True, activation="relu", dropout =
0.2)))
bilstm.add(Bidirectional(LSTM(32, return_sequences=True, activation="relu", dropout =
0.2)))
bilstm.add(Bidirectional(LSTM(32, return_sequences=True, activation="relu", dropout =
0.2)))
bilstm.add(Bidirectional(LSTM(32, activation = "relu", dropout = 0.2)))
bilstm.add(Dense(1, activation='sigmoid'))
opt = Adam(learning_rate=0.0001)
bilstm.compile(loss='binary_crossentropy', optimizer=opt,metrics=['accuracy'])
bilstm.summary()
bilstm.fit(train_padded, train_labels, epochs=20, batch_size=32,
validation_data=(valid_padded, valid_labels))
predictions_bilstm = bilstm.predict(test_padded)
predictions_bilstm = (predictions_bilstm > 0.5)
predictions_bilstm.astype('int64')
print(confusion_matrix(test_labels, predictions_bilstm))
print(classification_report(test_labels,predictions_bilstm))
print(accuracy_score(test_labels,predictions_bilstm))
from keras.layers import SimpleRNN
rnn = Sequential()
#First input and hidden layer
rnn.add(Embedding(num_words,32, input_length=max_length))
rnn.add(Dense(units=6, activation='relu'))
rnn.add(Dense(units=6, activation='relu'))
rnn.add(Dense(units=6, activation='relu'))
rnn.add(Dense(units=6, activation='relu'))
rnn.add(Dense(units=6, activation='relu'))
rnn.add(Flatten())
#Output layer
rnn.add(Dense(units=1))
#compiling
opt = Adam(learning_rate=0.001)
rnn.compile(optimizer = opt, loss = 'binary_crossentropy', metrics = ['accuracy'])
rnn.summary()
rnn.fit(train_padded, train_labels, epochs=60, batch_size=32,
validation_data=(valid_padded, valid_labels))
predictions_rnn = rnn.predict(test_padded)
predictions_rnn = (predictions_rnn > 0.5)
predictions_rnn.astype('int64')
print(confusion_matrix(test_labels, predictions_rnn))
print(classification_report(test_labels,predictions_rnn))
print(accuracy_score(test_labels,predictions_rnn))
```